2. Test cases

CRM API baseUrl: `https://cerebrumhubcrmlite.bubbleapps.io/ api/1.1`

Authorisation token: `b30ca93d0a7ebe389e9326a22497bf38`

Deals API

GET

Test case ID	**DG001: Get the list of all Deals**
Description	Verify that it is possible to retrieve the full list of all existing Deals.
Precondition	Some deals exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/deal` request
Expected result	• Successful code: 200 OK • List of all leads is returned in response

Test case ID	**DG002: Get the list of all Deals - no auth**
Description	Ensure that it is not possible to retrieve the list of Deals without being authorized.
Precondition	Some deals exist in the application, user is NOT authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/deal` request
Expected result	• Error code: 401 Unauthorized • Error message displayed: 'You do not have permission to get this object

Test case ID	**DG003: Get list of Deals- cursor 5, count 5**
Description	Verify that it is possible to retrieve the list of deals and set a valid limit value.
Precondition	Some deals exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/deal?cursor=5&count=5`
Expected result	• Successful code: 200 OK • List of all leads is returned in response, each batch of displayed deals has 10 of them • `count: 10`

Test case ID	**DG004: Get list of Deals- negative cursor**
Description	Check that using an invalid cursor value result in error.
Precondition	Some deals exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/deal?cursor=-10`
Expected result	• Successful code: 200 OK • Empty list is displayed • `cursor: -10`

Test case ID	**DG005: Get list of Deals- sort by company_name_text, ASC**
Description	Check that it is possible to sort Deals list by company name ASC.
Precondition	Some deals exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/deal?sort_field=company_name_text&boolean =false`
Expected result	• Successful code: 200 OK • List of all available deals is displayed, sorted by company name ASC

Test case ID	**DG006: Get list of Deals- sort by “Created Date”, ASC**
Description	Ensure, that it is possible to sort Deals list by created date ascending.
Precondition	Some deals exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/sort_field=Created Date&boolean=false`
Expected result	• Successful code: 200 OK • List of all available deals is displayed, sorted by “Created Date” ASC

Test case ID	**DG007: Get list of Deals- filtered by file_list_custom_data_file**
Description	Check that it is possible to filter out Deals without the files.
Precondition	Some deals exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/deal?constraints=[{"key"="file_list_custom_data_file","constraint_type":"not empty"}]`
Expected result	• Successful code: 200 OK • List of all available deals including files attached is displayed. • `file_list_custom_data_file` has values

Test case ID	**DG008: Get list of Deals- filtered by status_option_status "In progress", "Lost" and "Won"**
Description	Ensure, that it is possible to filter Deals by Status of the deal.
Precondition	Some deals exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/deal?constraints=[{"key"="status_option_status","constraint_type":"equals", "value":"In progress"}]` 2. Send **GET** : `{baseUrl}/obj/deal?constraints=[{"key"="status_option_status","constraint_type":"equals", "value":"Lost"}]` 3. Send **GET** : `{baseUrl}/obj/deal?constraints=[{"key"="status_option_status","constraint_type":"equals", "value":"Won"}]`
Expected result	1. Successful code: 200 OK • List of all available deals with status “In progress” is displayed. • `In progress` has values 2. Successful code: 200 OK • List of all available deals with status “Lost” is displayed. • `Lost` has values 3. Successful code: 200 OK • List of all available deals with status “Lost” is displayed. • `Won` has values

Test case ID	**DG009: Get list of Deals- filtered by created_by, DESC**
Description	Verify that it is possible to filter Deals by unique identifier of the user who added an object to the system; DESC.
Precondition	Some deals exist in the application, user is authorized.

Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/deal?constraints=[{"key"="created_by","constraint_type":"not empty"}]&boolean=true`
Expected result	• Successful code: 200 OK • List of all Deals by unique identifier (DESC) is displayed. • `created_by` has values

Test case ID	**DG010: Get list of Deals- filtered by logo_image**
Description	Check that it is possible to filter out Deals by company logos.
Precondition	Some deals exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/deal?constraints=[{"key"="logo_image","constraint_type":"not empty"}]`
Expected result	• Successful code: 200 OK • List of all available deals including company logos is displayed. • `logo_image` has values

GET by Id

Test case ID	**DGI001: Get Deal by Id**
Description	Verify that it is possible to retrieve an existing deal by its id.
Precondition	Some deals exist in the application, user is authorized
Test data	UniqueId = id of some existing deal
Test steps	1. Send **GET** : `{baseUrl}/obj/deal/:UniqueId` request
Expected result	• Successful code: 200 OK • Selected deal with inserted UniqueId is presented.

Test case ID	**DGI002: Get Deal by Id - no authorization**
Description	Ensure that it is not possible to retrieve an existing Deal by its id, without authorization.
Precondition	Some deals exist in the application, user is NOT authorized
Test data	UniqueId = id of some existing deal
Test steps	1. Send **GET** : `{baseUrl}/obj/deal/:UniqueId` request
Expected result	• Error code: 401 Unauthorized

POST

Test case ID	**DPO001: Create new deal with min field set**
Description	Verify that it is possible to create a new deal with min. (mandatory) field set
Precondition	User is authorized, user Id should exist in the system

Test data	<pre>{
	  "assignee__user": "1658145123067x157867491350963230",
	  "company_name_text": "Team 6 try by RiA",
	  "deal_value_estimation_number": 1,
	  "name_text": "Team 6 RiA test",
	  "order_number": 5,
	  "status_option_status": "",
	  "visible_to_list_user": []
	}</pre>

Test steps	1. Send **POST** `:{baseUrl}/obj/deal` request 2. Copy the “id” value from the response 3. Send **GET** : `{baseUrl}/obj/deal/:UniqueId` request
Expected result	• Successful code: 201 Created • Successful code: 200 OK, previously created deal is returned with proper values.

Test case ID	**DPO002: Create new deal with max field set**
Description	Verify that it is possible to create a new deal with max. field set (all fields)
Precondition	User is authorized, user ID, funnel, lead, note and stage exist in the system

Test data	<pre>{
	  "assignee__user": "1625047362532x398219546419355650",
	  "company_name_text": "{{$randomCompanyName}}",
	  "deal_value_estimation_number": 5000258,
	  "description_text": "{{$randomLoremSentence}}",
	  "file_list_custom_data_file": [],
	  "funnel_custom_funnel1": "1717009359123x720540828641325000",
	  "lead_custom_lead": "1717221446547x177524797604351680",
	  "logo_image": "https://media.macphun.com/img/uploads/customer
	  "name_text": "Team 6 try v. by RiA",
	  "note_list_custom_note": [
	  "1627393624856x199214596987551740"
	  ],
	  "order_number": 555,
	  "stage_custom_stage": "1625047818390x664413796313397600",
	  "status_option_status": "In progress",
	  "visible_to_list_user": [
	  "1625047362532x398219546419355650"
	  ]
	}</pre>

Test steps	1. Send **POST** `:{baseUrl}/obj/deal` request 2. Copy the “id” value from the response 3. Send **GET** : `{baseUrl}/obj/deal/:UniqueId` request
Expected result	• Successful code: 201 Created

	• Successful code: 200 OK, previously created deal is returned with proper values.

Test case ID	**DPO003: Create new deal with empty body**
Description	Verify that it is NOT possible to create a new deal with empty body
Precondition	User is authorized, user Id should exist in the system

Test data	{
	"assignee__user": "",
	"company_name_text": "",
	"deal_value_estimation_number": ,
	"description_text": "",
	"file_list_custom_data_file": "",
	"funnel_custom_funnel1": "",
	"lead_custom_lead": "",
	"logo_image": "",
	"name_text": "",
	"note_list_custom_note": [
	""
	],
	"order_number": ,
	"stage_custom_stage": "",
	"status_option_status": "",
	"visible_to_list_user": [
	""
	]
	}

Test steps	1. Send **POST** `:{baseUrl}/obj/deal` request
Expected result	• Error code: 400 Bad Request • Error message is displayed, explaining that is not possible create deal with empty body.

Test case ID	**DPO004: Create new deal without assignee_user**
Description	Verify that it is NOT possible to create a new deal without assignee_user
Precondition	User is authorized
Test data	{ "company_name_text": "{{$randomCompanyName}}", "deal_value_estimation_number": 607000, "description_text": "{{$randomLoremSentence}}", "file_list_custom_data_file": [ ], "logo_image": "", "funnel_custom_funnel1": "1717079423399x714608074658541300", "name_text": "Team 6 testing by RiA", "order_number": 0, "stage_custom_stage": "1625047829315x658646823454309500", "status_option_status": "In progress", "visible_to_list_user": ["1625125825767x718816571317085800"]

	16 } } } } } } } } } } } } } } } }
Test steps	1. Send **POST** `:{baseUrl}/obj/deal` request
Expected result	• Error code: 400 Bad Request • Error message is displayed, explaining that is not possible create deal without assignee_user.

Test case ID	**DPO005: Create new deal without company_name**
Description	Verify that it is NOT possible to create a new deal without company_name
Precondition	User is authorized, user Id should exist in the system
Test data	1 { 2 "assignee__user": "1625047362532x398219546419355650", 3 "deal_value_estimation_number": 607000, 4 "description_text": "{{$randomLoremSentence}}", 5 "file_list_custom_data_file": [6], 7 "logo_image": "", 8 "funnel_custom_funnel1": "1678718113594x500417408037027840", 9 "name_text": "Team6 try deals by RiA", 10 "order_number": 0, 11 "stage_custom_stage": "1625047829315x658646823454309500", 12 "status_option_status": "In progress", 13 "visible_to_list_user": [14 "1625047362532x398219546419355650" 15] 16 }
Test steps	1. Send **POST** `:{baseUrl}/obj/deal` request
Expected result	• Error code: 400 Bad Request • Error message is displayed, explaining what field is missing.

Test case ID	**DPO006: Create new deal without deal_value_estimation_number**
Description	Verify that it is NOT possible to create a new deal without deal_value_estimation number
Precondition	User is authorized, user Id should exist in the system
Test data	1 { 2 "assignee__user": "1625047362532x398219546419355650", 3 "company_name_text": "{{$randomCompanyName}}" 4 "description_text": "{{$randomLoremSentence}}", 5 "file_list_custom_data_file": [6], 7 "logo_image": "", 8 "funnel_custom_funnel1": "1678718113594x500417408037027840", 9 "name_text": "Team6 try deals by RiA", 10 "order_number": 0, 11 "stage_custom_stage": "1625047829315x658646823454309500", 12 "status_option_status": "In progress", 13 "visible_to_list_user": [14 "1625047362532x398219546419355650"

	<pre>15]
	16 }</pre>

Test steps	1. Send **POST** `:{baseUrl}/obj/deal` request
Expected result	• Error code: 400 Bad Request • Error message is displayed, explaining that is not possible create deal without deal_value_estimation_number.

PATCH

Test case ID	**DPA001: Modify the data of existing deal by assigne_user**
Description	Verify that it is possible to modify assignee_user of existing deal
Precondition	User is authorized, user Id and the deal should exist in the system

Test data	<pre>1 {
	2 "assignee__user": "1625047373081x778685405315052900"
	3 }</pre>

Test steps	1. Send **PATCH** `:{baseUrl}/obj/deal/:UnigueID` request 2. Send **GET** : `{baseUrl}/obj/deal/:UniqueId` request
Expected result	• Success code: 204. Deal data is updated. • Successful code: 200 OK, assignee field value has been updated successfully

Test case ID	**DPA002: Modify the data of existing deal by company_name_text**
Description	Verify that it is possible to modify company_name_text of existing deal
Precondition	User is authorized, user Id and the deal should exist in the system

Test data	<pre>1 {
	2 "company_name_text": "Team 6 try v by Ria"
	3 }</pre>

Test steps	1. Send **PATCH** `:{baseUrl}/obj/deal/:UnigueID` request 2. Send **GET** : `{baseUrl}/obj/deal/:UniqueId` request
Expected result	• Success code: 204. Deal data is updated. • Successful code: 200 OK, company_name_text field value has been updated successfully

Test case ID	**DPA003: Modify the data of existing deal by deal_value_estimation_number text**
Description	Verify that it is NOT possible to modify deal_value_estimation by text of existing deal
Precondition	User is authorized, user Id and the deal should exist in the system
Test data	<pre>1 {</pre>

	2 "deal_value_estimation_number": team six, 3 }
Test steps	1. Send **PATCH** `:{baseUrl}/obj/deal/:UnigueID` request
Expected result	• Error code: 400 Bad Request • Error message is displayed, explaining that is not possible use text in this field

Test case ID	**DPA004: Modify the data of existing deal by description_text number**
Description	Verify that it is NOT possible to modify description_text by number of existing deal
Precondition	User is authorized, user Id and the deal should exist in the system
Test data	1 { 2 "description_text": "458658974569441123614772563336541255 66", 3 }
Test steps	1. Send **PATCH** `:{baseUrl}/obj/deal/:UnigueID` request
Expected result	• Error code: 400 Bad Request • Error message is displayed, explaining that is not possible use number in this field

Test case ID	**DPA005: Modify the data of existing deal by lead_custom_lead**
Description	Verify that it is possible to modify lead_custom_lead of existing deal
Precondition	User is authorized, user Id and the deal should exist in the system
Test data	1 { 2 "lead_custom_lead": "1717221446547x177524797604351680", 3 }
Test steps	1. Send **PATCH** `:{baseUrl}/obj/deal/:UnigueID` request 2. Send **GET** : `{baseUrl}/obj/deal/:UniqueId` request
Expected result	• Success code: 204. Deal data is updated. • Successful code: 200 OK, lead_custom_lead field value has been updated successfully

Test case ID	**DPA006: Modify the data of existing deal by name_text**
Description	Verify that it is possible to modify name_text of existing deal
Precondition	User is authorized, user Id and the deal should exist in the system
Test data	1 { 2 "name_text": "Team 6 try v. by RiA", 3 }
Test steps	1. Send **PATCH** `:{baseUrl}/obj/deal/:UnigueID` request

	2. Send **GET** : `{baseUrl}/obj/deal/:UniqueId` request
Expected result	• Success code: 204. Deal data is updated. • Successful code: 200 OK, name_text field value has been updated successfully

Test case ID	**DPA007: Modify the data of non existing assigne_user**
Description	Verify that it is NOT possible to modify data of non existing assignee_user
Precondition	User is authorized, user Id and the deal should exist in the system
Test data	`{ "assignee__user": "1625047373081x778685405315054500" }`
Test steps	1. Send **PATCH** `:{baseUrl}/obj/deal/:UnigueID` request
Expected result	• Error code: 400 Bad Request • Error message is displayed, explaining that this assignee_user not exist in the system.

PUT

Test case ID	**DPU001: Replace the deal data with a new set of data**
Description	Verify that it is possible replace data of existing deal
Precondition	User is authorized, user Id and the deal should exist in the system
Test data	`{ "assignee__user": "1625047362532x398219546419355650", "company_name_text": "{{$randomCompanyName}}", "deal_value_estimation_number": 5000258, "description_text": "{{$randomLoremSentence}}", "file_list_custom_data_file": [], "funnel_custom_funnel1": "1717009359123x720540828641325000", "lead_custom_lead": "1717221446547x177524797604351680", "logo_image": "https://media.macphun.com/img/uploads/customer "name_text": "Team 6 try v. by RiA", "note_list_custom_note": ["1627393624856x199214596987551740"], "order_number": 555, "stage_custom_stage": "1625047818390x664413796313397600", "status_option_status": "In progress", "visible_to_list_user": ["1625047362532x398219546419355650"] }`
Test steps	1. Send **PUT** `:{baseUrl}/obj/:UnigueID` request 2. Send **GET** : `{baseUrl}/obj/deal/:UniqueId` request
Expected result	• Success code: 204. Deal data is replaced. • Successful code: 200 OK, deal data has been updated successfully.

Test case ID	**DPU002: Replace the deal data with empty body**
Description	Verify that it is possible replace existing deal data with empty body
Precondition	User is authorized, user Id and the deal should exist in the system

Test data	<pre>{
	    "assignee__user": null,
	    "company_name_text": null,
	    "deal_value_estimation_number": null,
	    "description_text": null,
	    "file_list_custom_data_file": [],
	    "funnel_custom_funnel1": null,
	    "lead_custom_lead": null,
	    "logo_image": null,
	    "name_text": null,
	    "note_list_custom_note": [],
	    "order_number": null,
	    "stage_custom_stage": null,
	    "status_option_status": null,
	    "visible_to_list_user": []
	}</pre>

Test steps	1. Send **PUT** `:{baseUrl}/obj/deal/:UnigueID` request 2. Send **GET** : `{baseUrl}/obj/deal/:UniqueId` request
Expected result	• Successful code: 204. Deal data is replaced. • Successful code: 200 OK, deal data has been updated successfully.

DELETE

Test case ID	**DDE001: Delete existing deal**
Description	Verify that it is possible delete existing deal
Precondition	User is authorized, user Id and the deal should exist in the system
Test data	UniqueID of existing deal
Test steps	1. Send **DELETE** `:{baseUrl}/obj/deal/:UnigueID` request 2. Send **GET** : `{baseUrl}/obj/deal/:UniqueId` request
Expected result	• Success code: 204. Deal is deleted • Error code: 404 Not Found; Error: `there is no such deal in the system` - delete action has been executed successfully.

Test case ID	**DDE002: Delete non existing deal**
Description	Verify that it is NOT possible delete existing deal
Precondition	User is authorized and the deal not exist in the system
Test data	UniqueID of non existing deal ex: `1625060551284x207376926164123650`

Test steps	1. Send **DELETE** `:{baseUrl}/obj/deal/:UnigueID` request
Expected result	• Error code: 404 Not Found • Error message is displayed: '`Missing object of type deal: object with id 1625060551284x207376926164123650 does not exist`

Notes API

GET

Test case ID	**NG001: Get the list of all Notes**
Description	It is possible to retrieve the full list of all existing Notes
Precondition	Some notes exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/note` request
Expected result	• Successful code: 200 OK • List of all leads is returned in response

Test case ID	**NG002: Get the list of all Notes - no auth**
Description	It is not possible to retrieve the list of Notes without being authorized
Precondition	Some Notes exist in the application, user is NOT authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/note` request
Expected result	• Error code: 401 Unauthorized • Error message displayed: ‘You do not have permission to get this object

Test case ID	**NG003: Get the list of all Notes- limit 7**
Description	Check if it is possible to filter out existing notes by their content.
Precondition	Some notes exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/note?limit=7` request
Expected result	• Successful code: 200 OK • List of seven leads is returned in response

Test case ID	**NG004: Get the list of all Notes- cursor 7**
Description	Verify that it is possible to retrieve the list of Leals and set a valid limit value.
Precondition	Some Notes exist in the application, user is authorized.

Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/note?cursor=7` request
Expected result	• Successful code: 200 OK • List of seven Notes is returned in response

Test case ID	**NG005: Get the list of all Notes- by sort_field Created Date**
Description	Check if it is possible to filter out existing Notes by sort_field = Created Date
Precondition	Some Notes exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/note?sort_field=Created Date` request
Expected result	• Successful code: 200 OK • List of Notes by Created Date is returned in response

Test case ID	**NG006: Get the list of all Notes- sort_field Modified Date, DESC**
Description	Check if it is possible to filter out existing notes by Modified Date, DESC.
Precondition	Some Notes exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/note?sort_field=Modified Date&descending=true` request
Expected result	• Successful code: 200 OK • List of Notes by Modified Date (DESC) is returned in response

Test case ID	**NG007: Get the list of all Notes- sort_field Created By**
Description	Check if it is possible to filter out existing notes by Created By.
Precondition	Some Notes exist in the application, user is authorized.
Test data	-
Test steps	1. Send **GET** : `{baseUrl}/obj/note?sort_field=Created By` request
Expected result	• Successful code: 200 OK • List of Notes by Created By is returned in response

Test case ID	**NG008: Get the list of all Notes- "content_text" constraint**
Description	Check if it is possible to filter out existing notes by their content.
Precondition	Some Notes exist in the application, user is authorized.
Test data	`"Offer a new deal to the client"`

Test steps	1. Send **GET** : `{baseUrl}/obj/note?constraints=[{"key": "content_text", "constraint_type": "equals", "value": "Offer a new deal to the client"}]` request
Expected result	• Successful code: 200 OK • Note(s) containing `"Offer a new deal to the client"` is/are returned in response

GET by Id

Test case ID	**NGI001: Get Note by Id**
Description	It is possible to retrieve an existing Note by its id.
Precondition	Some Notes exist in the application, user is authorized
Test data	UniqueId = id of some existing Note
Test steps	1. Send **GET** : `{baseUrl}/obj/note/:UniqueId` request
Expected result	• Successful code: 200 OK • Selected Note with inserted UniqueId is presented.

Test case ID	**NGI002: Get Note by Id - no authorization**
Description	It is not possible to retrieve an existing Note by its id, without authorization.
Precondition	Some Notes exist in the application, user is NOT authorized
Test data	UniqueId = id of some existing Note
Test steps	1. Enter (key) UniqueId: 1717686121819x421995122994541100 2. Send **GET** : `{baseUrl}/obj/note/:UniqueId` request
Expected result	• Error code: 401 Unauthorized

POST

Test case ID	**NPO001: Create a new Note**
Description	Check of it is possible to create a note with text value.
Precondition	user is authorized
Test data	`"Still Extremely good job Team6, yes it is indeed"`
Test steps	1. Send **POST** : `{baseUrl}/obj/note` request 2. Copy the `"id"` field value from the response 3. Send **GET** : `{baseUrl}/obj/note/:UniqueId`
Expected result	Successful code: 201 Created Step 3: Successful code: 200 OK; correct data of the previously created note is displayed

Test case ID	**NPO002: Create a new Note- random text**
Description	Verify that it is possible to create a note with a random text value
Precondition	user is authorized
Test data	`{ "content_text": "{{$randomPhrase}}" }`
Test steps	1. Send **POST** request to URL: `{{baseUrl}}/obj/note` 2. Copy the `id` field value from the response 3. Send **GET** : `{baseUrl}/obj/note/:UniqueId`
Expected result	Step 1: Successful code: 201 Created Step 3: Successful code: 200 OK; correct data of the previously created note is displayed

Test case ID	**NPO003: Create a new Note- empty body**
Description	Verify that user is not allowed to create a Note with an empty request body.
Precondition	user is authorized
Test data	`{ "content_text": "" }`
Test steps	1. Send **POST** request to URL: `{{baseUrl}}/obj/note`
Expected result	• Error code: 400 Bad Request • Informative error message is displayed, explaining which field is missing

PATCH

Test case ID	**NPA001-PATCH a Note by ID**
Description	Try to modify a note by unique ID
Precondition	User is authorized, note with valid id is existing
Test data	Unique_id nr: `1717686121819x421995122994541100` `{ "content_text": "FunnelTunnelRoadRunner" }`
Test steps	1. Paste UniqueID nr into value textbox in folder : **PATCH** a note by ID 2. Change “content_text” field value in BODY 3. Send **PATCH** `:{baseUrl}/obj/note/:UnigueID` request
Expected result	• Shown message: Status 204 Success. Note replaced • User could modify a note by ID

Test case ID	**NPA002-PATCH a Note by ID- no authorization**
Description	Try to modify a note by unique ID without authorization
Precondition	User is NOT authorized, note with valid id is existing
Test data	Unique_id nr: `1717686121819x421995122994541100` `{ "content_text": "FunnelTunnelRoadRunner" }`
Test steps	1. Paste UniqueID nr into value textbox in folder : **PATCH** a note by ID 2. Change “content_text” field value in BODY 3. Send **PATCH** `:{baseUrl}/obj/note/:UnigueID` request
Expected result	• Error code: 401 Unauthorized

Test case ID	**NPA003-Negative1-PATCH a note by invalid ID**
Description	Try to **PATCH** a note by invalid ID
Precondition	User is authorized
Test data	`{ "content_text": "FunnelTunnelRoadRunner" }`
Test steps	1. Enter invalid UniqueID value into folder: **PATCH** a note by ID 2. Send **PATCH** `:{baseUrl}/obj/note/:UnigueID` request
Expected result	Error message is shown 404 Not found

Test case ID	**NPA004- PATCH a note by id- boolean**
Description	Check, that it is NOT possible to modify content of an existing note with boolean value
Precondition	User is authorized, some notes exist
Test data	`{ "content_text": false }`
Test steps	1. Send **PATCH** `:{baseUrl}/obj/note/:UnigueID` request
Expected result	• Error code: 400 Bad Request • Informative error message `"Invalid data for field content_text: Expected a string, but got a boolean (original data: false)"` is displayed

Test case ID	**NPA005- PATCH a note by id- numeric**
Description	Check, that it is NOT possible to modify content of an existing note with numeric value

Precondition	User is authorized, some notes exist
Test data	`{ "content_text": 08062024 }`
Test steps	1. Send **PATCH** `:{baseUrl}/obj/note/:UnigueID` request
Expected result	• Error code: 400 Bad Request • Informative error message `Error: Error parsing request body: Unexpected number in JSON at position 22` is displayed

Test case ID	**NPA006- PATCH a note by id- empty body**
Description	Check, that it is NOT possible to modify content of an empty request body
Precondition	User is authorized, some notes exist
Test data	`{}`
Test steps	1. Send **PATCH** `:{baseUrl}/obj/note/:UnigueID` request
Expected result	• Error code: 400 Bad Request • Informative error message is displayed, explaining which field is missing

Test case ID	**NPA007- PATCH a note by id- empty content**
Description	Check, that it is NOT possible to modify content of an empty content field
Precondition	User is authorized, some notes exist
Test data	`{ "content_text": null }`
Test steps	1. Send **PATCH** `:{baseUrl}/obj/note/:UnigueID` request
Expected result	• Error code: 400 Bad Request • Informative error message is displayed, explaining which field is missing

PUT

Test case ID	**NPU001-PUT a note by Unique_id**
Description	Try to replace a note with a new stage by unique ID
Precondition	User is authorized
Test data	Unique_id: `1717686121819x421995122994541100` `{ "content_text": "FunnelTunnelRoadRunner" }`
Test steps	1. Copy an Unique Id of the note that you are going to replace

	2. Paste copied unique_id nr into **PUT** folder 3. Change “content_text” field value in BODY etc `LoaderRoaderFunnelTunnel` 4. Send **PUT** `:{baseUrl}/obj/note/:UnigueID` request
Expected result	• Shown message: Status 204 Success. Note replaced • User can Replace a note with a new stage by unique ID

Test case ID	**NPU002 Negative-PUT a note by empty Unique_id**
Description	Try to replace a note with a new stage by empty unique ID
Precondition	User is authorized

Test data	Unique_id: `“.....................”`
	{
	"content_text": "EmptyTunnelRunner"
	}

Test steps	1. Leve Unique Id value box empty that you are going to replace 2. Change “content_text” field value in BODY etc `EmptyTunnelRunner` 3. Send **PUT** `:{baseUrl}/obj/note/:UnigueID` request
Expected result	Status Error message 400 Database retrieval failure

Test case ID	**NPU003 Negative-PUT a unauthorized note by Unique_id**
Description	Try to replace unauthorized note with a new stage by unique ID
Precondition	User is unauthorized in the main system

Test data	Unique_id: `“`1717663491689x475531168506445800`"`
	{
	"content_text": "EmptyTunnelRunner"
	}

Test steps	1. Get some uniqunique_id from GET Note, e.g 1717663491689x475531168506445800 2. Paste Unique Id into value box in folder PUT Note new stage by Unique_id 3. Change AuthorAuthorization value to “ Not auth” 4. Send **PUT** `:{baseUrl}/obj/note/:UnigueID` request
Expected result	• Showing message Status Error message 401 Unauthorized • `"You do not have permission to modify this object"`

DELETE

Test case ID	**NDE001-DELETE a note by uniqunique_id**
Description	Deletes a thing of type note by unique ID
Precondition	User is authorized

Test data	Unique Id of the note that you are going to delete eg `1717680430315x946423265631648900`
Test steps	1. Copy an Unique Id of the note that you are going to DELETE 2. Paste copied unique_id nr into DELETE folder 3. Send `:{baseUrl}/obj/note/:UnigueID` request
Expected result	• Shown message: Status 204 Success. Note deleted • Deletes a selected note by unique ID

Test case ID	**NDE002 Negative-DELETE a note by invalid unique_id**
Description	Try to delete a note by invalid unique_id
Precondition	User is authorized
Test data	-
Test steps	1. Enter an invalid Unique Id into value box 2. Send **DELETE** `:{baseUrl}/obj/note/:UnigueID` request
Expected result	• Error message is shown: 400 Database retrieval failure • User can’t delete a note by invalid unique_id