	Steps	ER
Prerequisites	1. Open terminal on your computer	Window with terminal opened successfully
	2. Download docker image via: docker pull azshoo/alaska:1.0	Docker file appears in computer
	3. Run docker using: docker run -it -p 9091:8091 --name trololo azshoo/alas	Application opened with message: "Alaska - ========== ALASKA =========="
	Important! All next commands for checking must be executed in another tab of terminal.	
Testcase	Steps	ER
Testcase 1. Show options	1. Send command curl -X get http://127.0.0.1:9091/info	Command line shows available options for user: Welcome to Alaska! This is CRUD service for bears in alaska. CRUD routes presented with REST naming notation: POST /bear - create GET /bear - read all bears GET /bear/:id - read specific bear PUT /bear/:id - update specific bear DELETE /bear - delete all bears DELETE /bear/:id - delete specific bear Example of ber json: {"bear_type":"BLACK","bear_name":"mikhail","bear_age":17.5}. Available types for bears are: POLAR, BROWN, BLACK and GUMMY
Testcase 2. Create bear	1. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"POLAR","bear_name":"IGOR"," bear_age":'99'}' http://127.0.0.1:9091/bear	Command line returns id of new bear as integer value
	2. Save/remember returned bear_id	
	3. Send command curl -X GET http://127.0.0.1:9091/bear/{bear_id}	Command line returns json structure of created bear that consist of "bear_type", "bear_name", "bear_age" and "bear_id"
	4. Check that all values of fields of returned bear as the same like created bear	All fields must be the same that was set in create operation

Testcase 3. Delete bear	1. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"BLACK","bear_name":"POUL"," bear_age":'99'}' http://127.0.0.1:9091/bear	Command line returns id of new bear as integer value
	2. Save/remember returned bear_id	
	3. Send command curl -X DELETE http://127.0.0.1:9091/bear/{bear_id}	Command line returns 'OK' message
	4. Send command curl -X GET http://127.0.0.1:9091/bear/{bear_id}	Command line returns 'EMPTY' message
Testcase 4. Show bears	1. Send commands curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"BLACK","bear_name":"PETR"," bear_age":'12'}' http://127.0.0.1:9091/bear curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"BROWN","bear_name":"GLEB"," bear_age":'3'}' http://127.0.0.1:9091/bear curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"POLAR","bear_name":"ZAHAR"," bear_age":'9'}' http://127.0.0.1:9091/bear curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"GUMMY","bear_name":"SERJ"," bear_age":'11'}' http://127.0.0.1:9091/bear one by one	Command line returns id of new bear as integer value after each command
	2. Send command curl -X GET http://127.0.0.1:9091/bear	Command line returns list with json structures of each created bear that consist of "bear_type", "bear_name", "bear_age" and "bear_id"
	3. Check that all values of fields of returned bears as the same like created bears	All fields must be the same that was set in create operation
	1. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"BLACK","bear_name":"LU"," bear_age":'10'}' http://127.0.0.1:9091/bear	Command line returns id of new bear as integer value

Testcase 5. Update bear	2. Save/remember returned bear_id	
	3. Send command curl -X GET http://127.0.0.1:9091/bear/{bear_id}	Command line returns json structure of created bear that consist of "bear_type", "bear_name", "bear_age" and "bear_id"
	4. Check that all values of fields of returned bear as the same like created bear	All fields must be the same that was set in create operation
	5. Send command curl -X PUT -H "Content-Type: application/json" -d '{"bear_type":"POLAR","bear_name":"ALEX","bear_age":'100'}' http://127.0.0.1:9091/bear/{bear_id}	Command line returns 'OK' message
	6. Send command curl -X GET http://127.0.0.1:9091/bear/{bear_id}	Command line returns json structure of updated bear that consist of "bear_type", "bear_name", "bear_age" and "bear_id"
	7. Check that all values of fields of returned bear as the same like bear from Step (5)	All fields values must be the same that was set in update operation
Testcase 6. Delete bears	1. Send commands curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"BLACK","bear_name":"FEDOR","bear_age":'19'}' http://127.0.0.1:9091/bear curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"BROWN","bear_name":"IGNAT","bear_age":'33'}' http://127.0.0.1:9091/bear curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"POLAR","bear_name":"OLEG","bear_age":'97'}' http://127.0.0.1:9091/bear curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"GUMMY","bear_name":"FILL","bear_age":'1'}' http://127.0.0.1:9091/bear one by one	Command line returns id of new bear as integer value after each command
	2. Send command curl -X GET http://127.0.0.1:9091/bear	Command line returns list with json structures of each created bear that consist of "bear_type", "bear_name", "bear_age" and "bear_id"
	3. Check that all values of fields of returned bears as the same like created bears	All fields must be the same that was set in create operation
	4. Send command curl -X DELETE http://127.0.0.1:9091/bear	Command line returns 'OK' message

	5. Send command curl -X GET http://127.0.0.1:9091/bear	Command line returns '[]' (empty list)
Testcase 7. Create bear with empty data	1. Send command for create bear without any data curl -X POST -H "Content-Type: application/json" -d '{}' http://127.0.0.1:9091/bear	Command line returns 'Error. Pls fill all parameters' message
Testcase 8. Create bear with empty value in fields	1. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"","bear_name":"","bear_age":''}' http://127.0.0.1:9091/bear	Command line returns '500 Internal Server Error'
	2. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"BLACK","bear_name":"","bear_age":''}' http://127.0.0.1:9091/bear	Command line returns '500 Internal Server Error'
	3. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"BLACK","bear_name":"BOB"," bear_age":''}' http://127.0.0.1:9091/bear	Command line returns '500 Internal Server Error'
	4. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"","bear_name":"","bear_age":'77'}' http://127.0.0.1:9091/bear	Command line returns '500 Internal Server Error'
	5. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"","bear_name":"BOB","bear_age":'77'}' http://127.0.0.1:9091/bear	Command line returns '500 Internal Server Error'
	6. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"","bear_name":"BOB","bear_age":''}' http://127.0.0.1:9091/bear	Command line returns '500 Internal Server Error'

Testcase 9. Create bear with wrong type	1. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"PANDA","bear_name":"PO"," bear_age":'7'}' http://127.0.0.1:9091/bear	Command line returns '500 Internal Server Error'
Testcase 10. Create bear with to many fields	1. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"POLAR","bear_name":"ZAK"," bear_age":'3',"bear_job":"CIRCUS"}' http://127.0.0.1:9091/bear	Command line returns '500 Internal Server Error'
Testcase 11. Create bear with specific age	1. Send command with each {invalid_value} from list: [0, -1, a, A, ц, Ц, $, ?] curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"BLACK","bear_name":"ROB"," bear_age":{invalid_value}}' http://127.0.0.1:9091/bear	Command line returns '500 Internal Server Error'
Testcase 12. Delete non-existing bear	1. Send command curl -X GET http://127.0.0.1:9091/bear	Command line returns all existing bears
	2. Save/remember bear_id that free and does't exists in any bear	
	3. Send command curl -X DELETE http://127.0.0.1:9091/bear/{bear_id}	Command line returns 'EMPTY' message
Testcase 13. Delete bear by specific id	1. Send command with each {invalid_value} from list: [0, -1, a, A, ц, Ц, $, ?] curl -X DELETE http://127.0.0.1:9091/bear/ {invalid_value}	Command line returns '500 Internal Server Error'
Testcase 13. Show non-existing bear	1. Send command curl -X GET http://127.0.0.1:9091/bear	Command line returns all existing bears
	2. Save/remember bear_id that free and does't exists in any bear	
	3. Send command curl -X GET http://127.0.0.1:9091/bear/{bear_id}	Command line returns 'EMPTY' message

Testcase 14. Delete bear by specific id	1. Send command with each {invalid_value} from list: [0, -1, a, A, ц, Ц, $, ?] curl -X GET http://127.0.0.1:9091/bear/{invalid_value}	Command line returns '500 Internal Server Error'
Testcase 15. Update bear by not all fields	1. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"BLACK","bear_name":"LU"," bear_age":'10'}' http://127.0.0.1:9091/bear	Command line returns id of new bear as integer value
	2. Save/remember returned bear_id	
	3. Send command curl -X GET http://127.0.0.1:9091/bear/{bear_id}	Command line returns json structure of created bear that consist of "bear_type", "bear_name", "bear_age" and "bear_id"
	4. Check that all values of fields of returned bear as the same like created bear	All fields must be the same that was set in create operation
	5. Send command curl -X PUT -H "Content-Type: application/json" -d '{"bear_type":"POLAR"}' http://127.0.0.1:9091/bear/{bear_id}	Command line returns 'OK' message
	6. Send command curl -X GET http://127.0.0.1:9091/bear/{bear_id}	Command line returns json structure of updated bear that consist of "bear_type", "bear_name", "bear_age" and "bear_id"
	7. Check that all values of "bear_type" of returned bear as the same like was set in Step (5)	A field value must be the same that was set in update operation
	8. Send command curl -X PUT -H "Content-Type: application/json" -d '{"bear_name":"TIM"}' http://127.0.0.1:9091/bear/{bear_id}	Command line returns 'OK' message
	9. Send command curl -X GET http://127.0.0.1:9091/bear/{bear_id}	Command line returns json structure of updated bear that consist of "bear_type", "bear_name", "bear_age" and "bear_id"
	10. Check that all values of "bear_name" of returned bear as the same like was set in Step (8)	A field value must be the same that was set in update operation
	11. Send command curl -X PUT -H "Content-Type: application/json" -d '{"bear_age":"33"}' http://127.0.0.1:9091/bear/{bear_id}	Command line returns 'OK' message

	12. Send command curl -X GET http://127.0.0.1:9091/bear/{bear_id}	Command line returns json structure of updated bear that consist of "bear_type", "bear_name", "bear_age" and "bear_id"
	13. Check that all values of "bear_age" of returned bear as the same like was set in Step (11)	A field value must be the same that was set in update operation
Testcase 16. Update non-existing bear	1. Send command curl -X GET http://127.0.0.1:9091/bear	Command line returns all existing bears
	2. Save/remember bear_id that free and does't exists in any bear	
	3. Send command curl -X PUT -H "Content-Type: application/json" -d '{"bear_type":"POLAR","bear_name":"ALEX"," bear_age":'100'}' http://127.0.0.1:9091/bear/{bear_id}	Command line returns '500 Internal Server Error'
Testcase 17. Show non-existing animals	1. Send command curl -X GET http://127.0.0.1:9091/fox	Command line returns '404 Not found'
Testcase 18. Create bear with dublicate fields	1. Send command curl -X POST -H "Content-Type: application/json" -d '{"bear_type":"BROWN","bear_type":"POLAR"," bear_name":"ALEX","bear_name":"FRED"," bear_age":'10',"bear_age":"44"}' http://127.0.0.1:9091/bear	Command line returns id of new bear as integer value
	2. Save/remember returned bear_id	
	3. Send command curl -X GET http://127.0.0.1:9091/bear/{bear_id}	Command line returns json structure of created bear that consist of "bear_type", "bear_name", "bear_age" and "bear_id"
	4. Check that all values of fields of returned bear as the same like second value of each field	All fields must be like second value in create operation