REST API REPORT

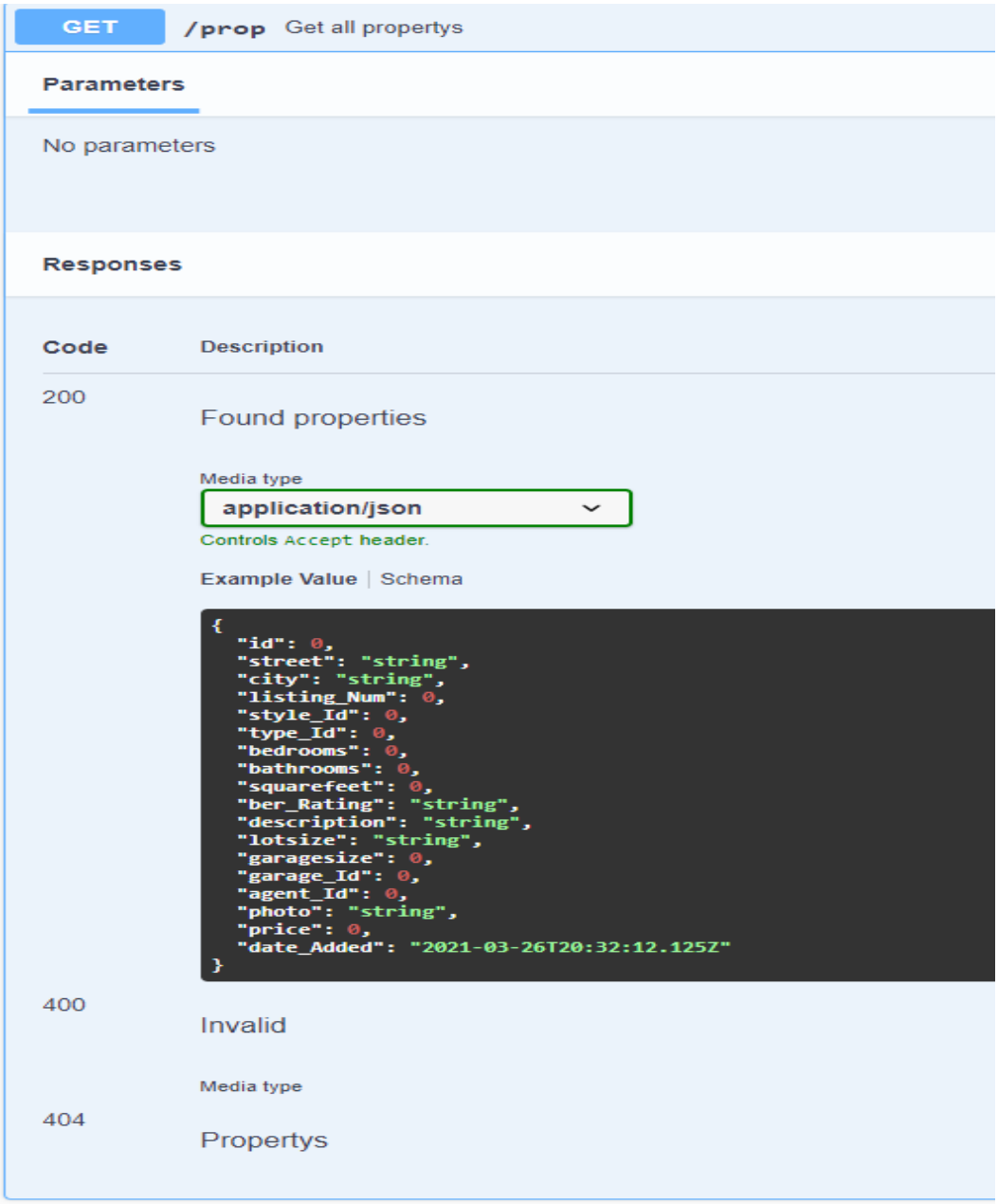
K00227507: Andrew Lennox

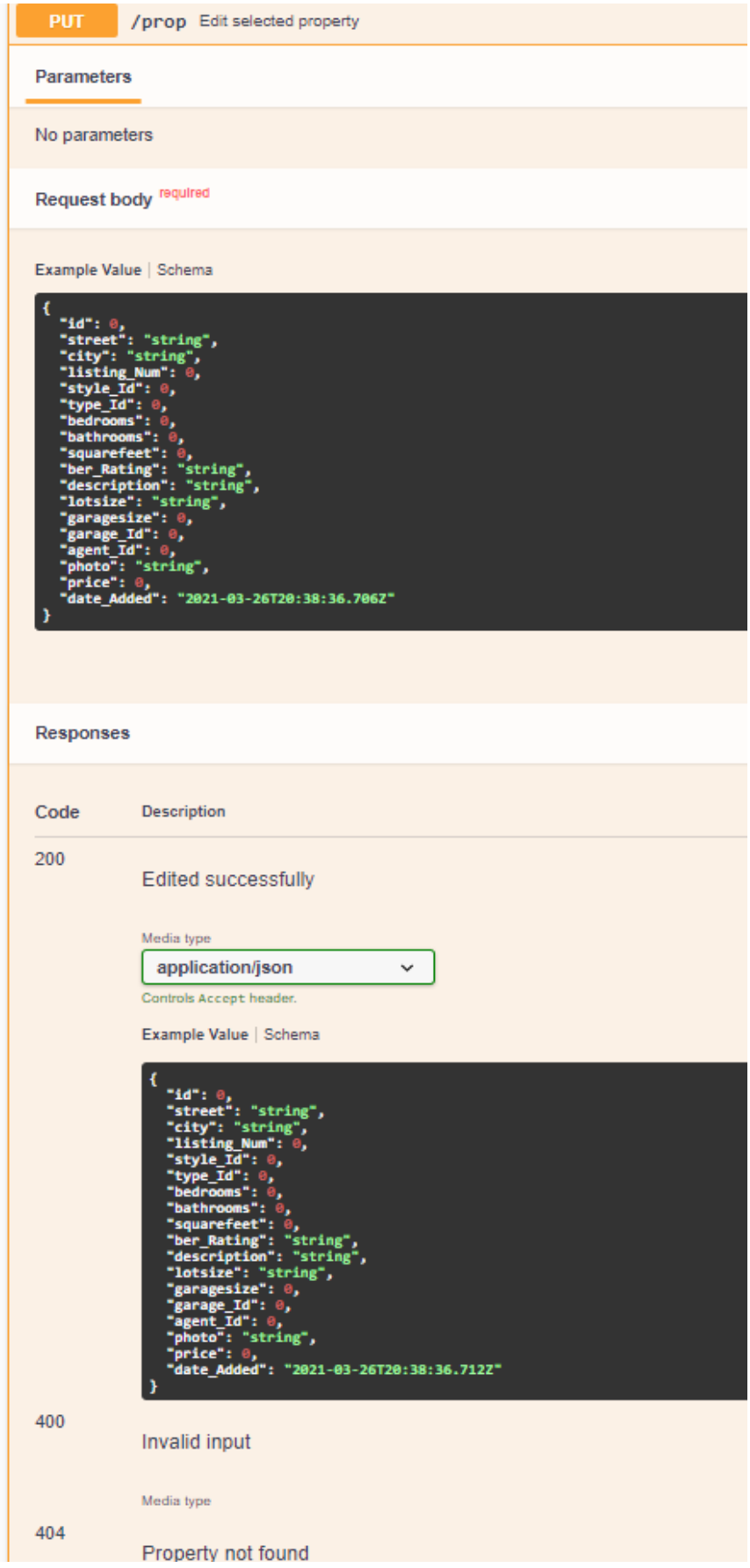
Table of Contents

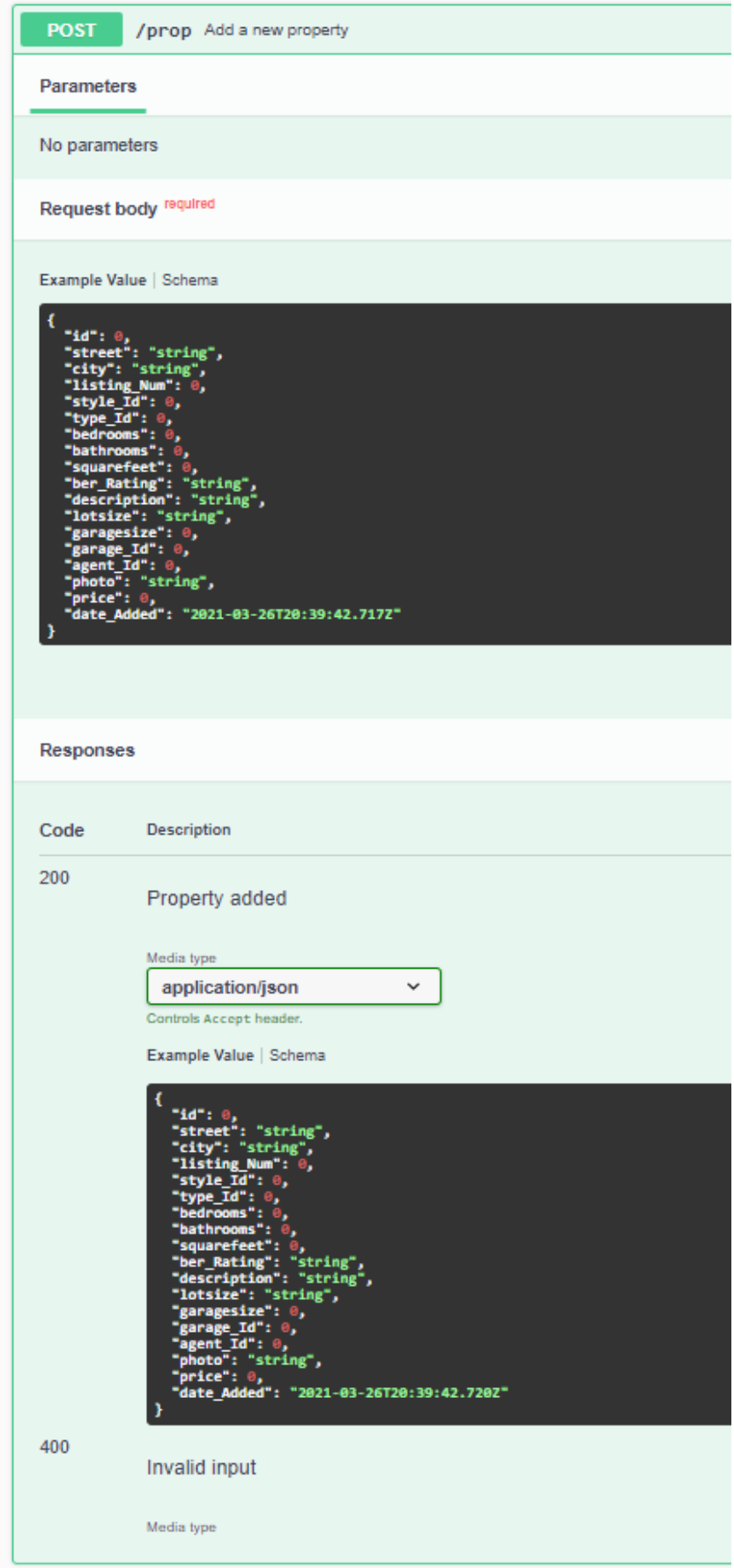
NOTE: For the API Documentation I have provided screenshots of each function in the documentation I created with Swagger. However, this documentation can also be viewed upon start-up of project at the following url:

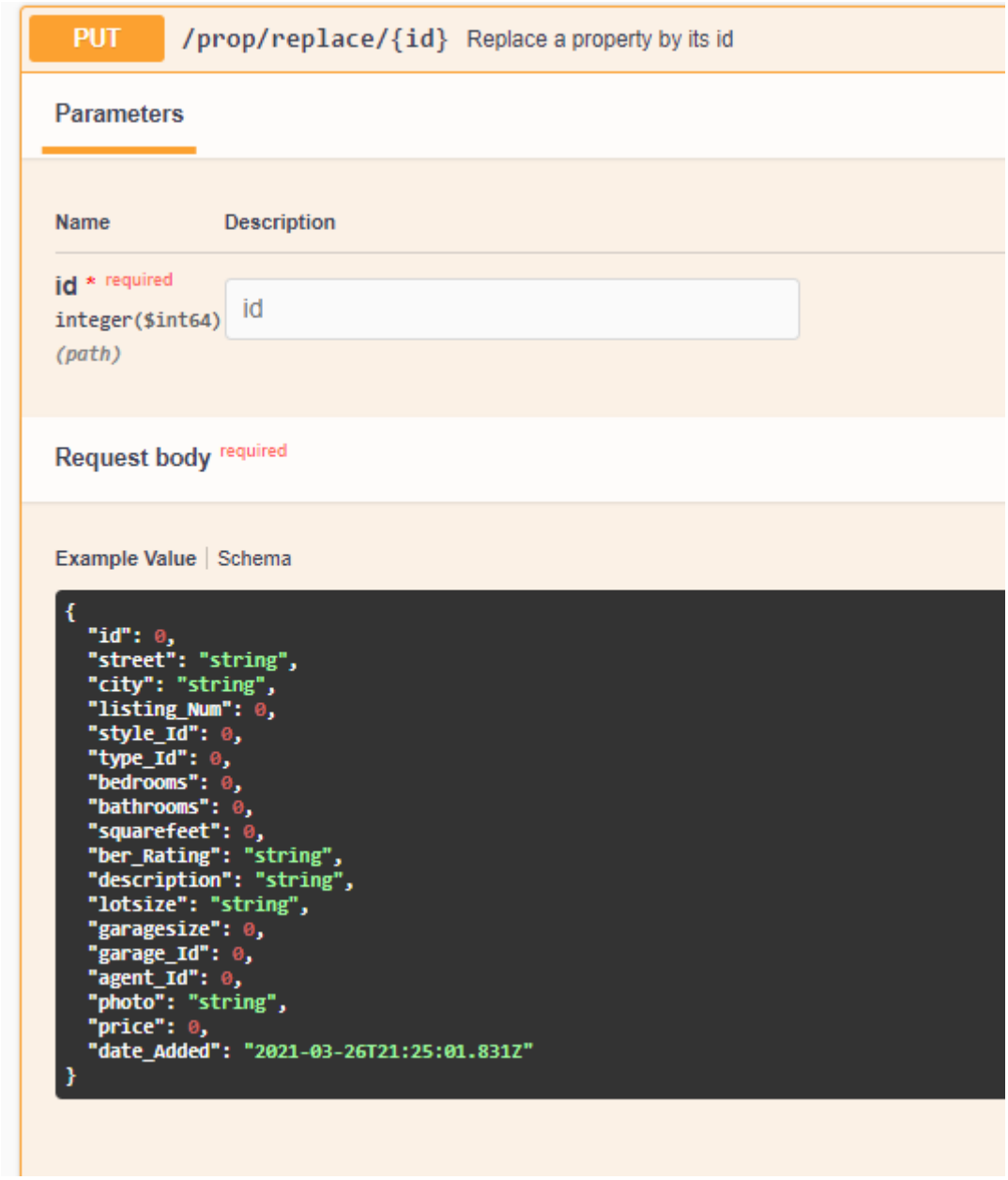
http://localhost:8080/swagger-ui.html

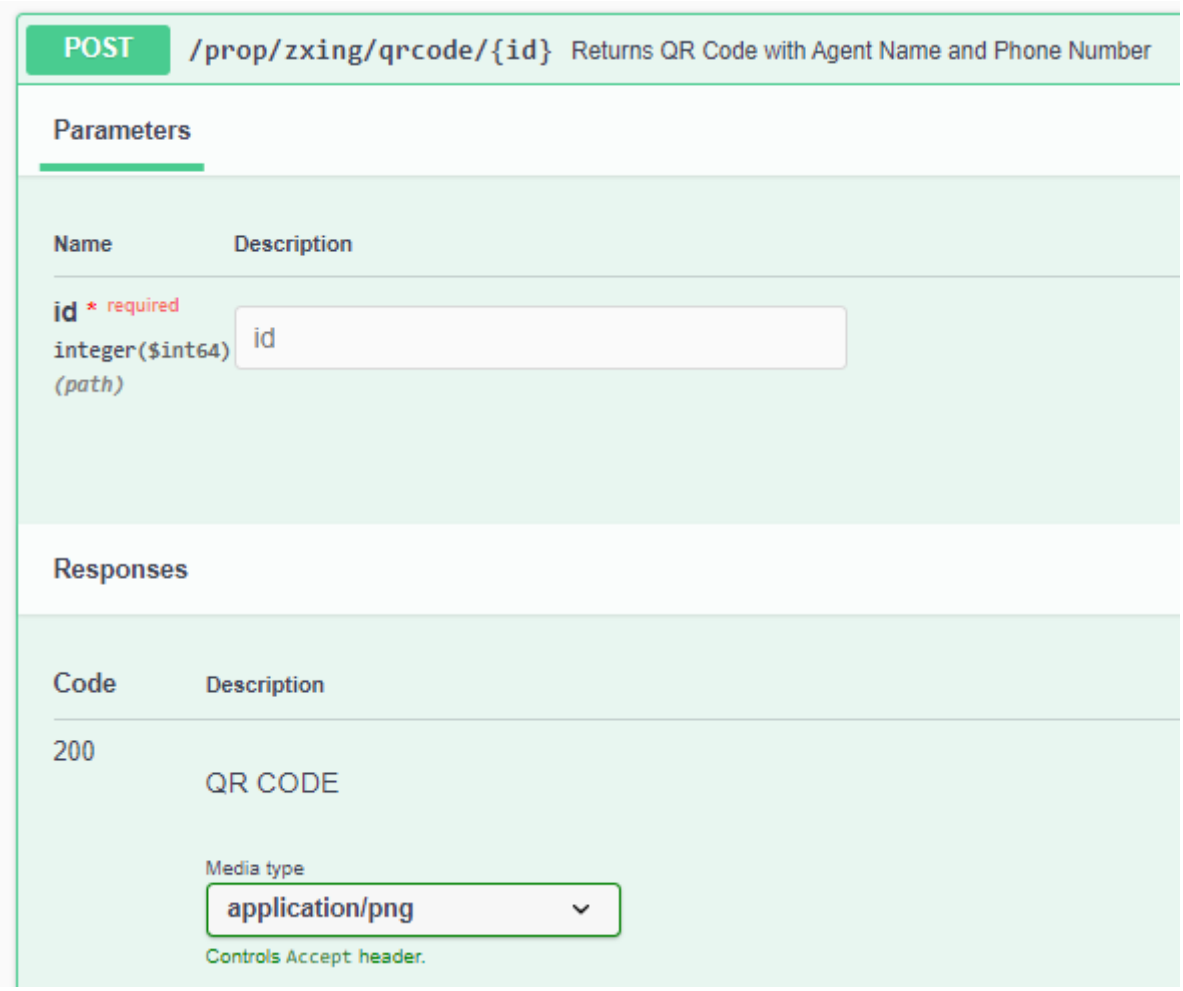
1 API DOCUMENTATION

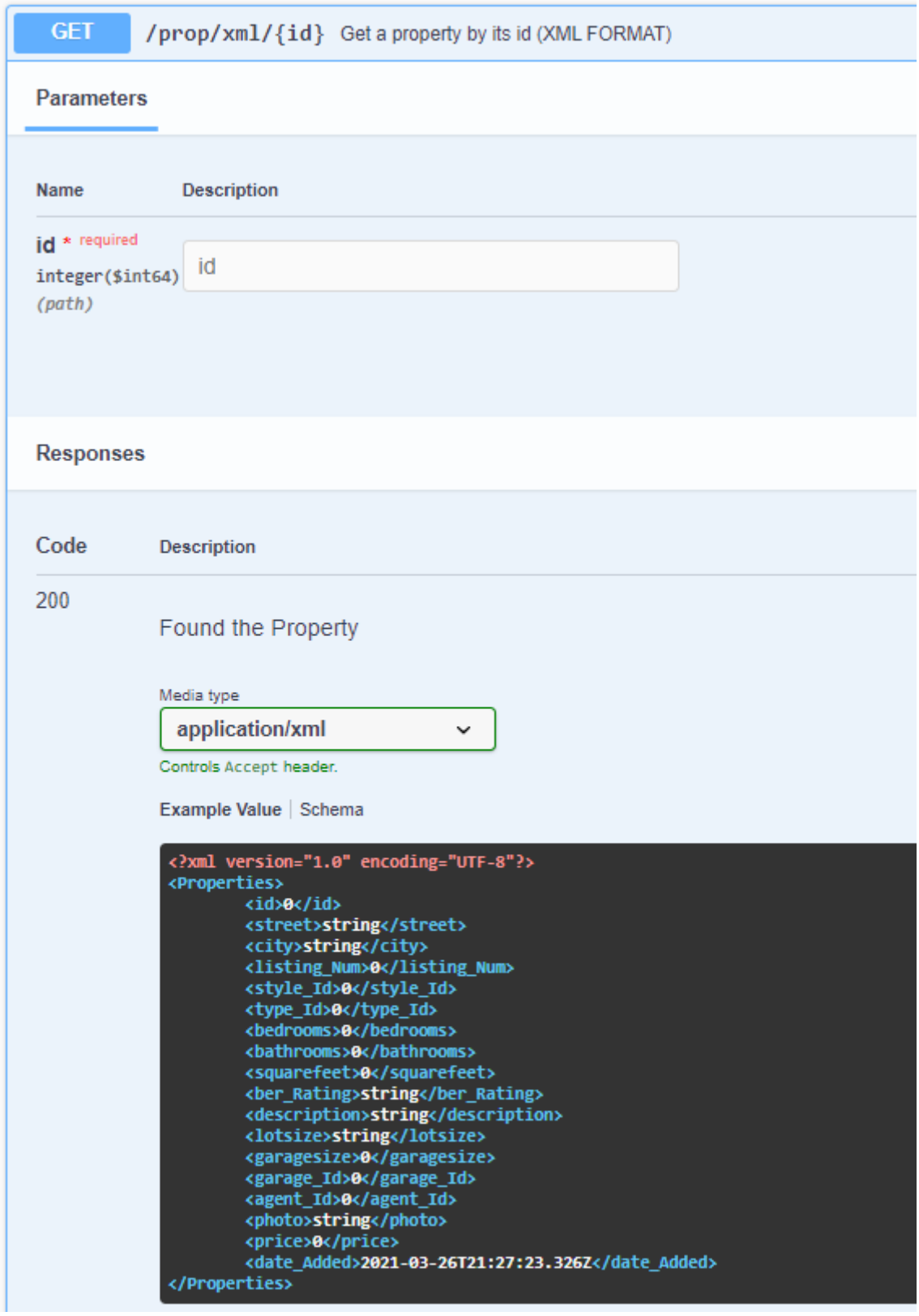


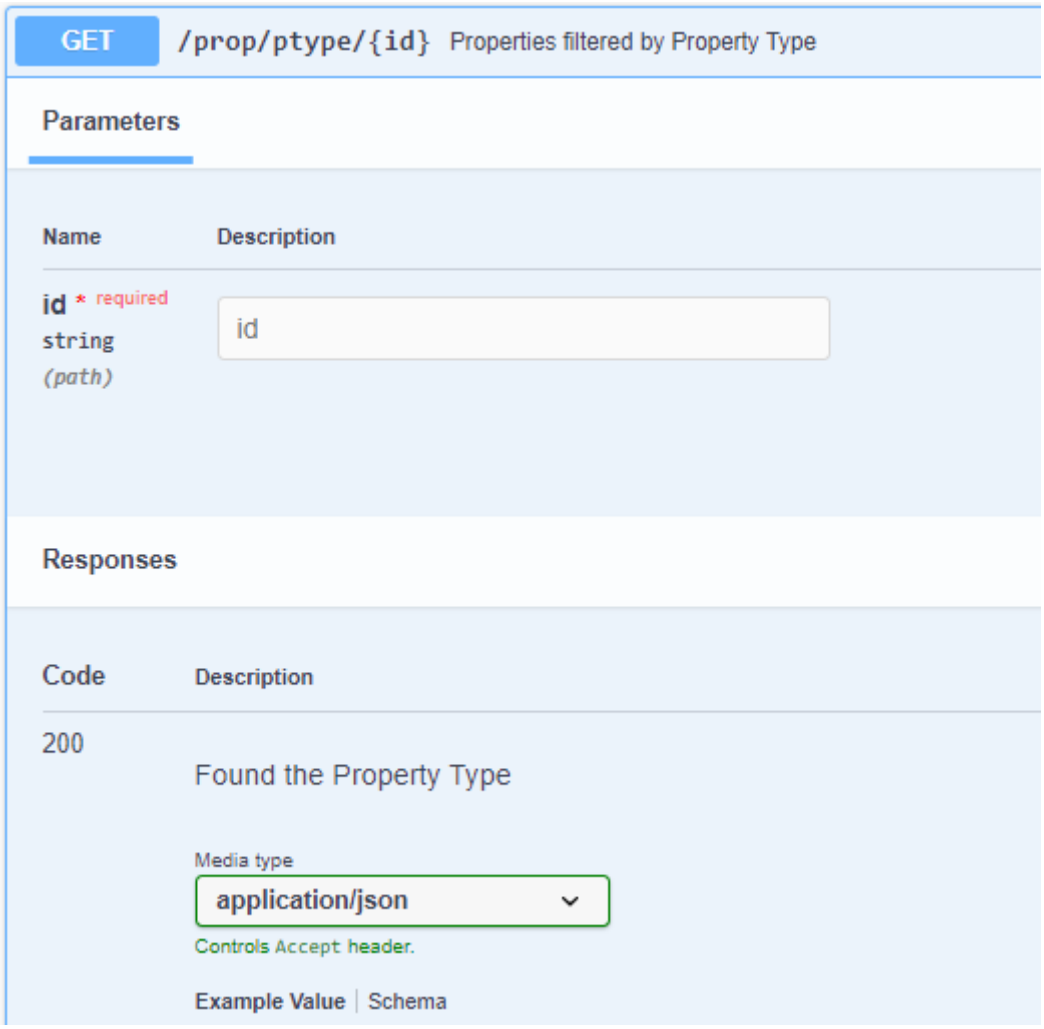

PUT
/prop Edit selected property
Parameters
No parameters
Request body required
Example Value | Schema
{
"id": 0,
"street": "string",
"city": "string",
"listing_Num": 0,
"style_Id": 0,
"type_Id": 0,
"bedrooms": 0,
"bathrooms": 0,
"squarefeet": 0,
"ber_Rating": "string",
"description": "string",
"lotsize": "string",
"garagesize": 0,
"garage_Id": 0,
"agent_Id": 0,
"photo": "string",
"price": 0,
"date_Added": "2021-03-26T20:38:36.706Z"
}
Responses
Code
Description
200
Edited successfully
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"street": "string",
"city": "string",
"listing_Num": 0,
"style_Id": 0,
"type_Id": 0,
"bedrooms": 0,
"bathrooms": 0,
"squarefeet": 0,
"ber_Rating": "string",
"description": "string",
"lotsize": "string",
"garagesize": 0,
"garage_Id": 0,
"agent_Id": 0,
"photo": "string",
"price": 0,
"date_Added": "2021-03-26T20:38:36.712Z"
}
400
Invalid input
Media type
404
Property not found

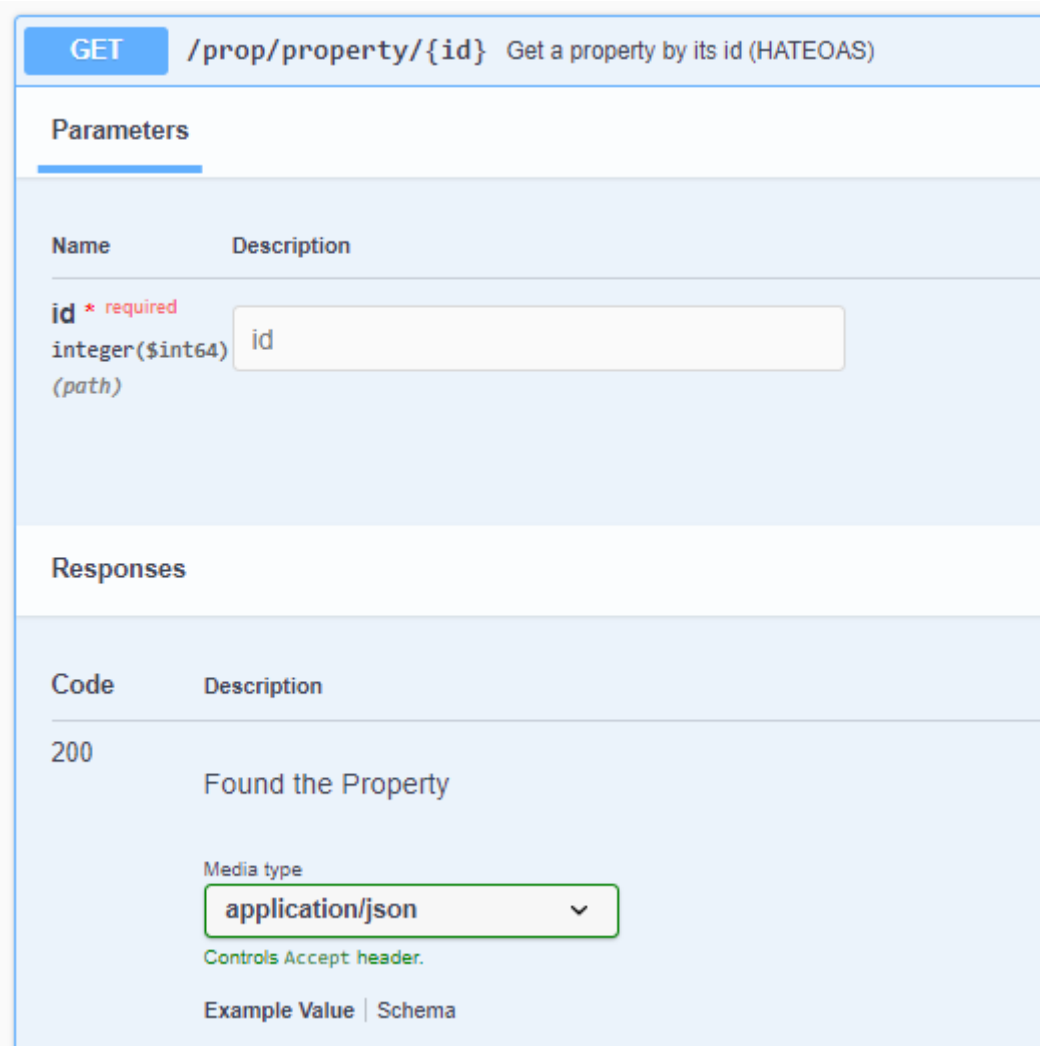

POST
/prop
Add a new property
Parameters
No parameters
Request body
required
Example Value | Schema
{
"id": 0,
"street": "string",
"city": "string",
"listing_Num": 0,
"style_Id": 0,
"type_Id": 0,
"bedrooms": 0,
"bathrooms": 0,
"squarefeet": 0,
"ber_Rating": "string",
"description": "string",
"lotsize": "string",
"garagesize": 0,
"garage_Id": 0,
"agent_Id": 0,
"photo": "string",
"price": 0,
"date_Added": "2021-03-26T20:39:42.717Z"
}
Responses
Code
Description
200
Property added
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"street": "string",
"city": "string",
"listing_Num": 0,
"style_Id": 0,
"type_Id": 0,
"bedrooms": 0,
"bathrooms": 0,
"squarefeet": 0,
"ber_Rating": "string",
"description": "string",
"lotsize": "string",
"garagesize": 0,
"garage_Id": 0,
"agent_Id": 0,
"photo": "string",
"price": 0,
"date_Added": "2021-03-26T20:39:42.720Z"
}
400
Invalid input
Media type

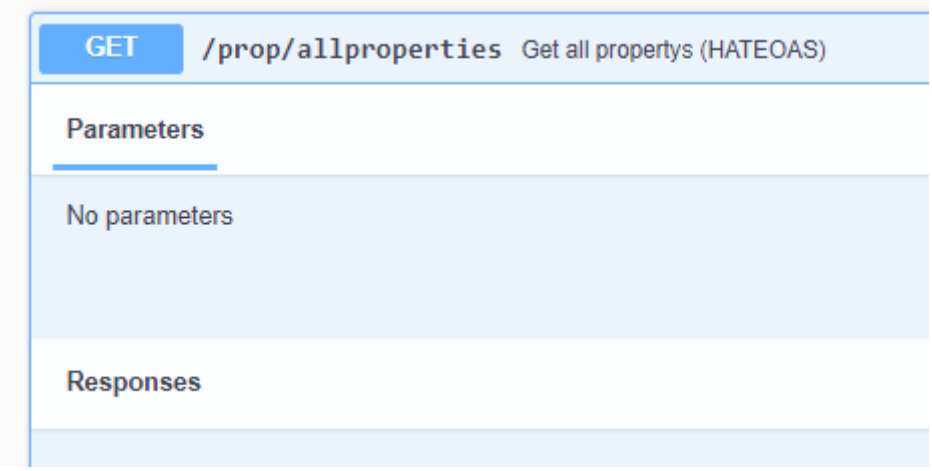

PUT
/prop/replace/{id}
Replace a property by its id
Parameters
Name
Description
id * required
integer($int64)
(path)
id
Request body required
Example Value | Schema
{
"id": 0,
"street": "string",
"city": "string",
"listing_Num": 0,
"style_Id": 0,
"type_Id": 0,
"bedrooms": 0,
"bathrooms": 0,
"squarefeet": 0,
"ber_Rating": "string",
"description": "string",
"lotsize": "string",
"garagesize": 0,
"garage_Id": 0,
"agent_Id": 0,
"photo": "string",
"price": 0,
"date_Added": "2021-03-26T21:25:01.831Z"
}

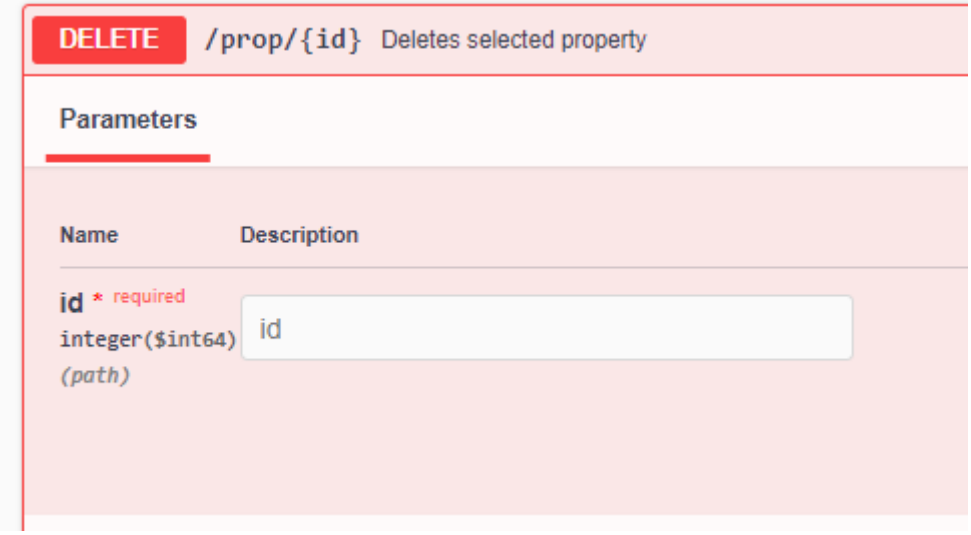

POST
/prop/zxing/qrcode/{id}
Returns QR Code with Agent Name and Phone Number
Parameters
Name
Description
id * required
integer($int64)
(path)
id
Responses
Code
Description
200
QR CODE
Media type
application/png
Controls Accept header.

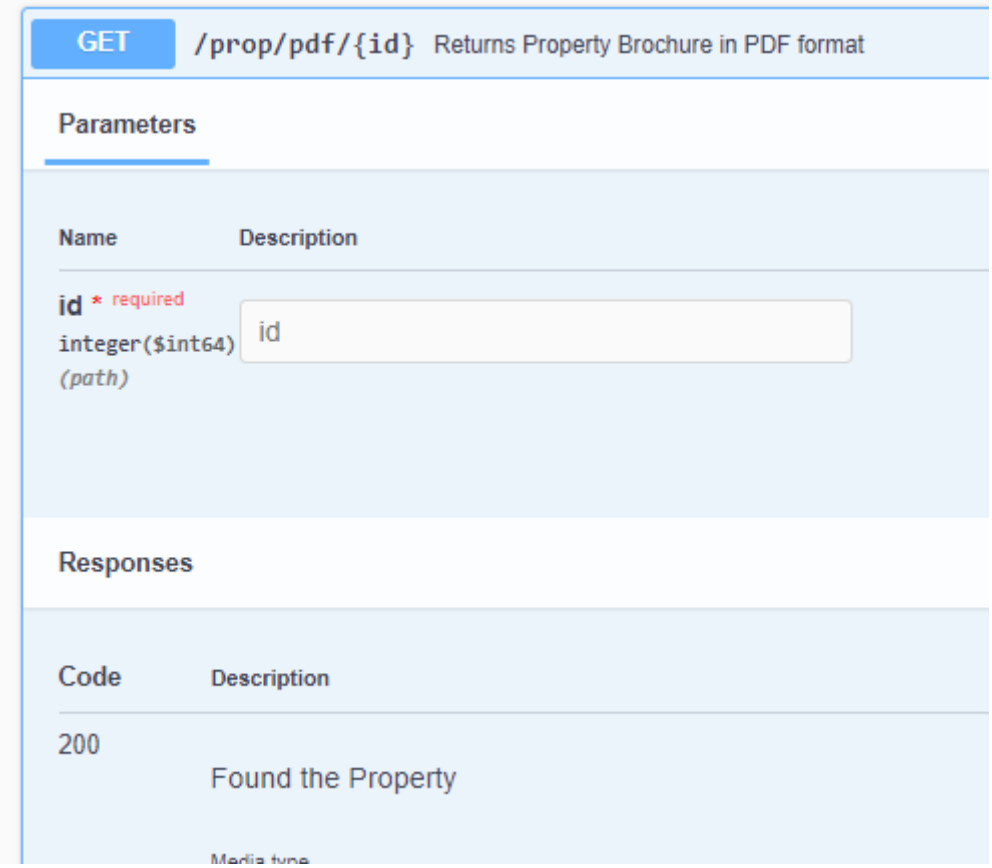

GET
/prop/xml/{id} Get a property by its id (XML FORMAT)
Parameters
Name
Description
id * required
integer($int64)
(path)
id
Responses
Code
Description
200
Found the Property
Media type
application/xml
Controls Accept header.
Example Value | Schema
<?xml version="1.0" encoding="UTF-8"?>
<Properties>
<id>0</id>
<street>string</street>
<city>string</city>
<listing_Num>0</listing_Num>
<style_Id>0</style_Id>
<type_Id>0</type_Id>
<bedrooms>0</bedrooms>
<bathrooms>0</bathrooms>
<squarefeet>0</squarefeet>
<ber_Rating>string</ber_Rating>
<description>string</description>
<lotsize>string</lotsize>
<garagesize>0</garagesize>
<garage_Id>0</garage_Id>
<agent_Id>0</agent_Id>
<photo>string</photo>
<price>0</price>
<date_Added>2021-03-26T21:27:23.326Z</date_Added>
</Properties>


GET
/prop/ptype/{id}
Properties filtered by Property Type
Parameters
Name
Description
id * required
string
(path)
id
Responses
Code
Description
200
Found the Property Type
Media type
application/json
Controls Accept header.
Example Value
Schema


GET
/prop/property/{id}
Get a property by its id (HATEOAS)
Parameters
Name
Description
id * required
integer($int64)
(path)
id
Responses
Code
Description
200
Found the Property
Media type
application/json
Controls Accept header.
Example Value
Schema


GET
/prop/allproperties
Get all propertys (HATEOAS)
Parameters
No parameters
Responses


DELETE
/prop/{id}
Deletes selected property
Parameters
Name
Description
id * required
integer($int64)
(path)
id



Example return:

Property Brochure

City : Befast Street : 88 Lagmore Glen

Lovely home in a great neighborhood. Plenty of space. Private back yard. With your design flair, this could be your showpiece home.

Lot Size	Bathroom	Bedrooms	Garage Size
80x110	2.0	3	1

Price : €200800.0

© LIT Realty

GET /prop/getimage/{mm}/{id} Get a property Image

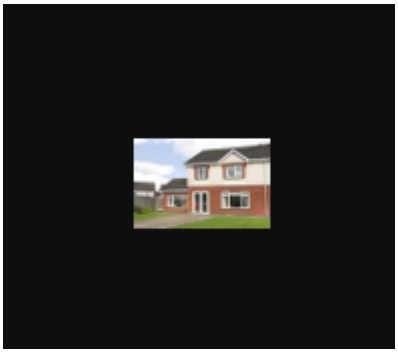
Parameters

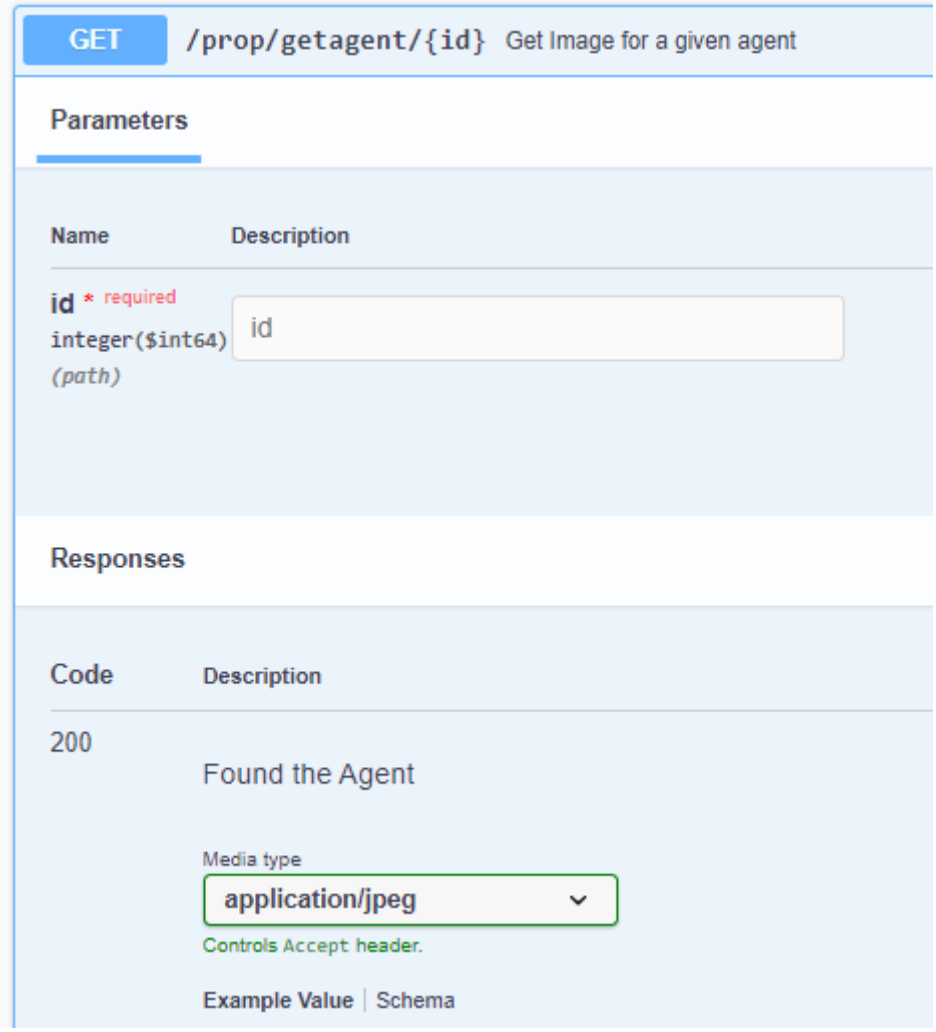
Name	Description
mm * required string *(path)*	mm
id * required integer($int64) *(path)*	id

Responses

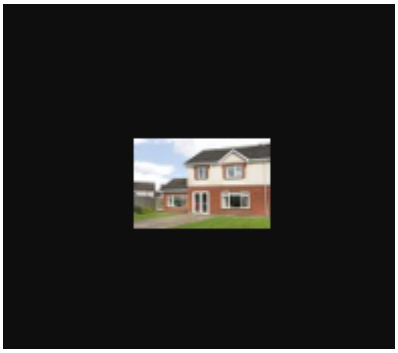
Code	Description
200	Found the Property Media type application/jpeg Controls Accept header.

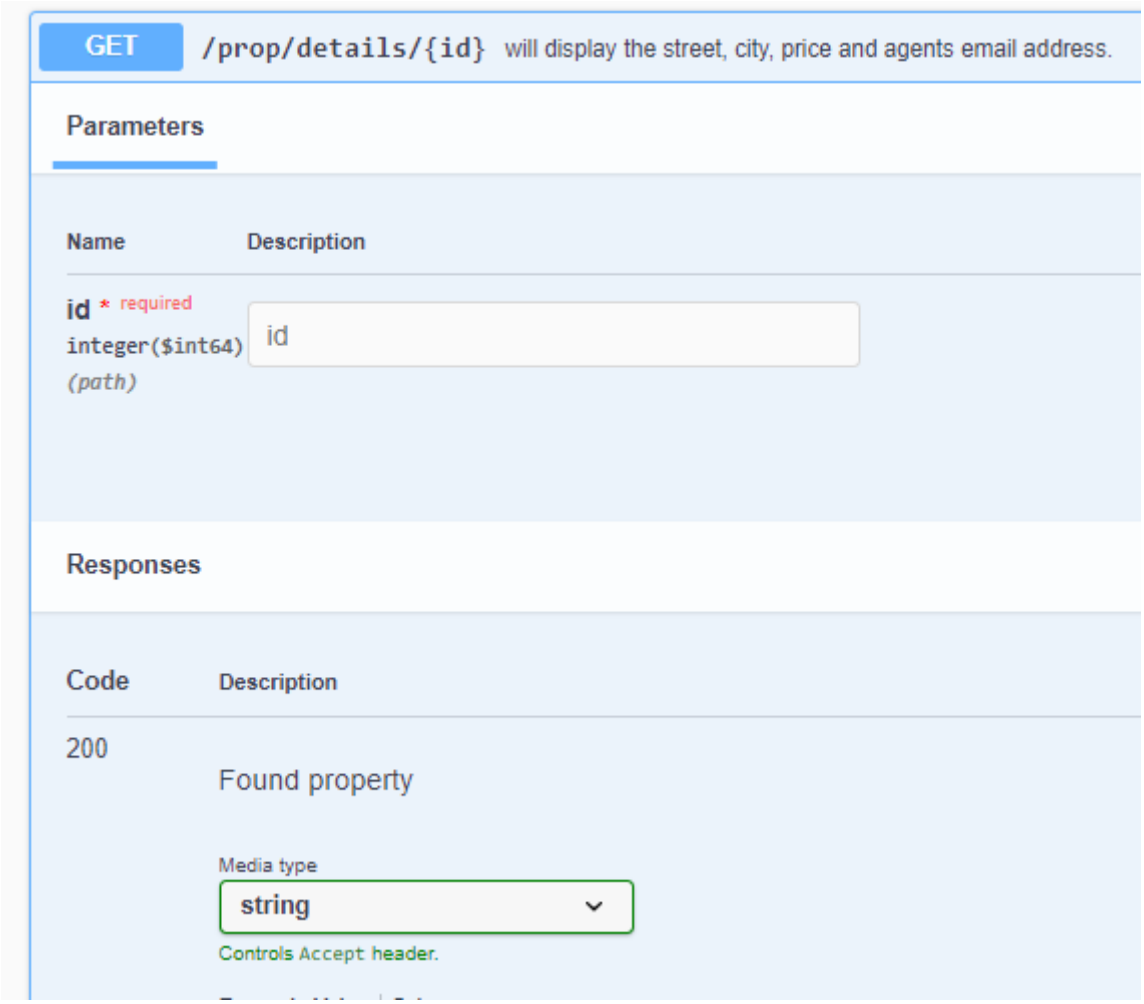
Example return:

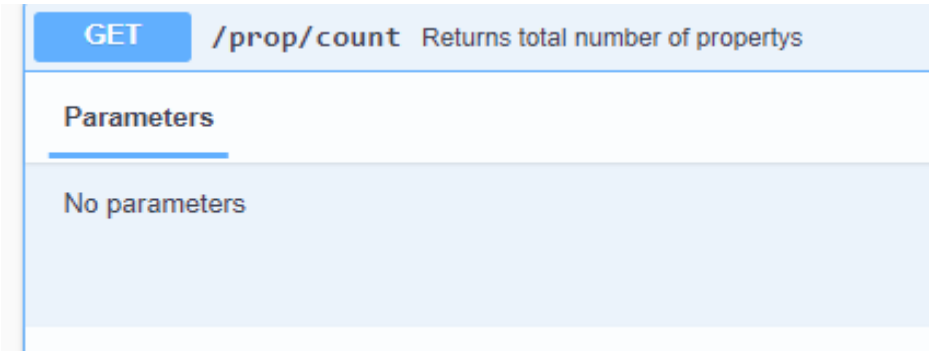


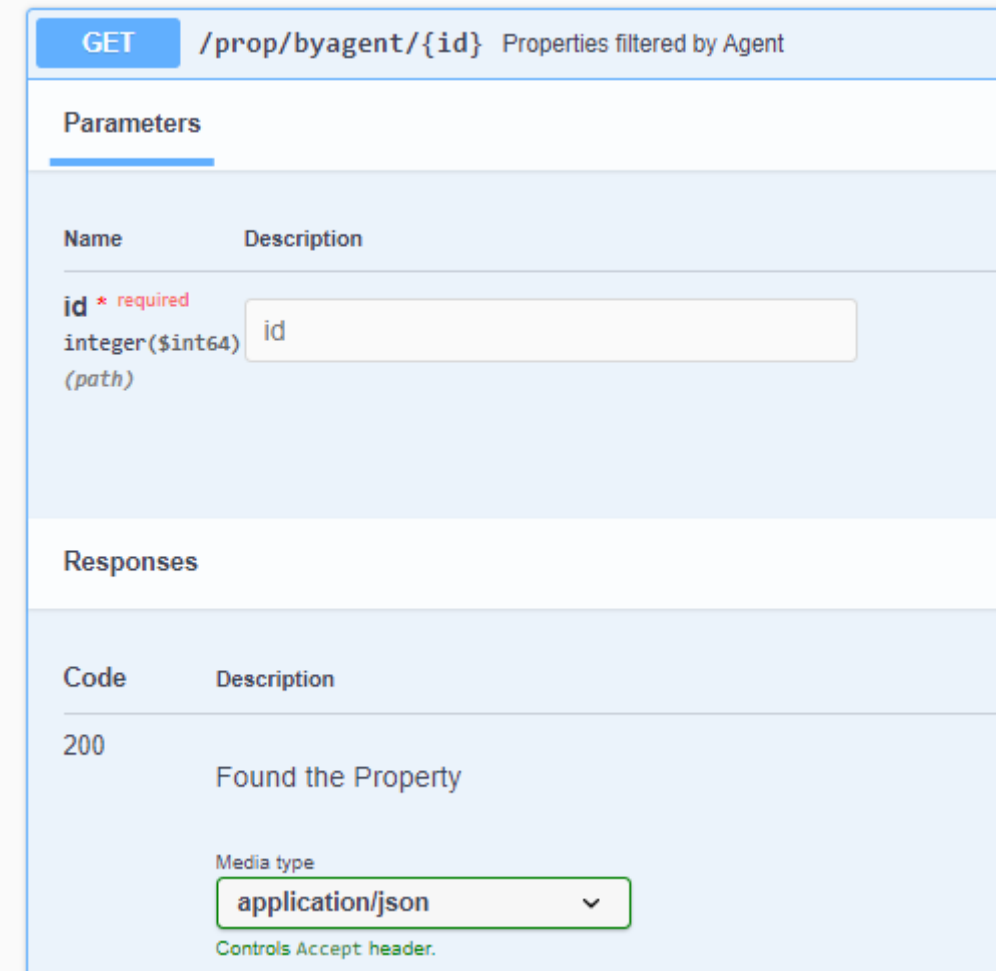


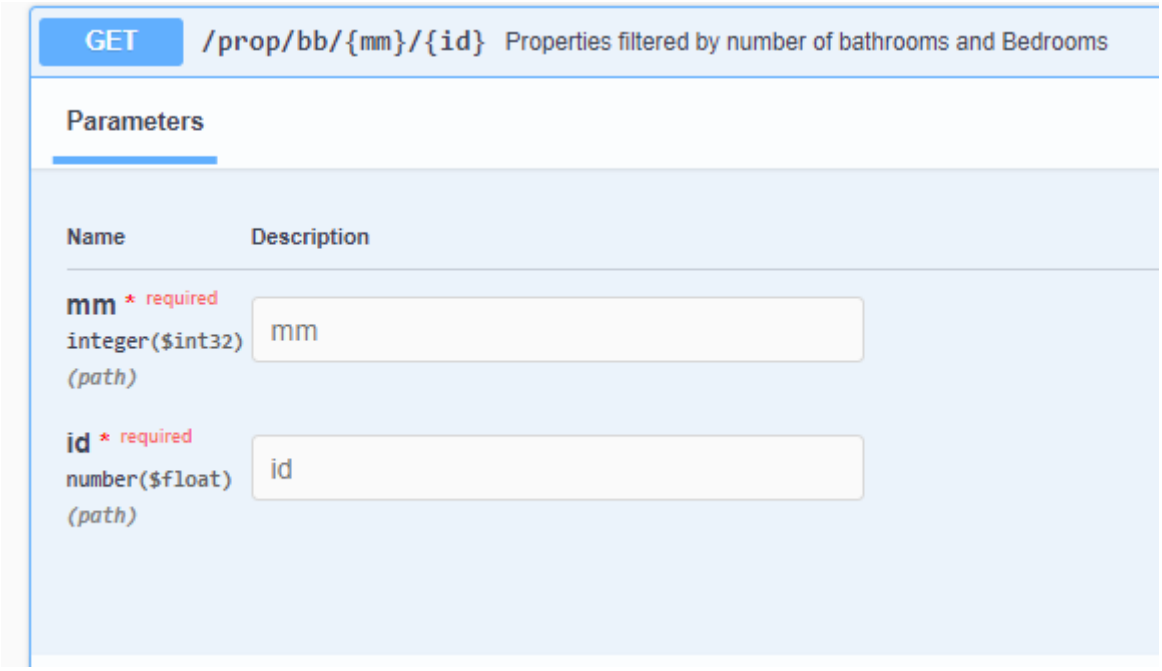
Example return:

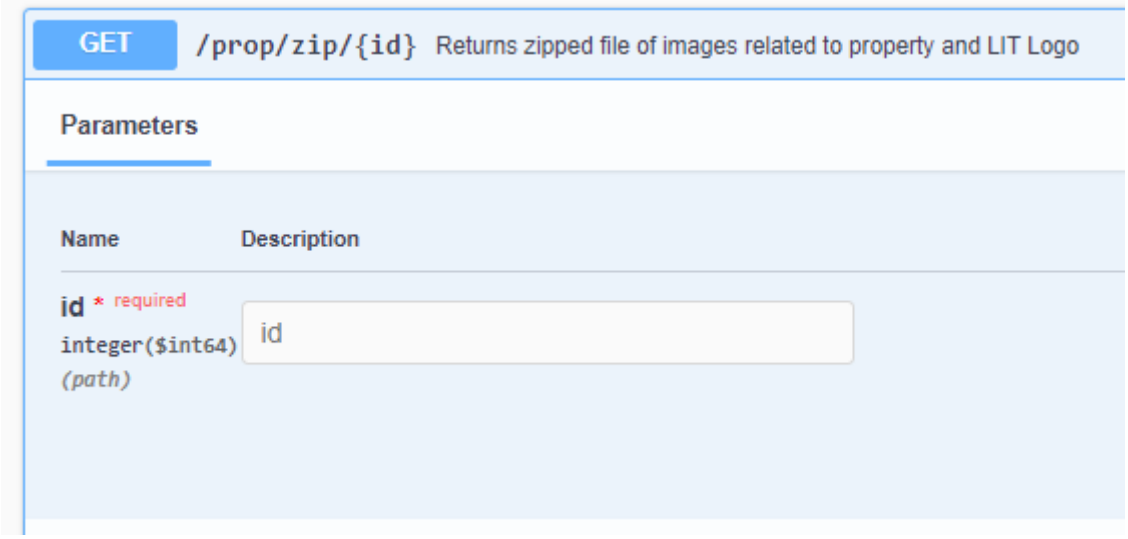



GET
/prop/details/{id} will display the street, city, price and agents email address.
Parameters
Name
Description
id * required
integer($int64)
(path)
id
Responses
Code
Description
200
Found property
Media type
string
Controls Accept header.

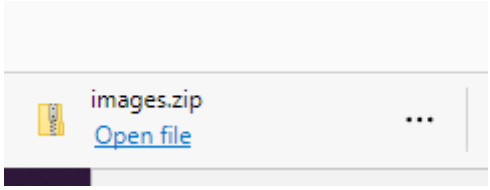

GET
/prop/count Returns total number of propertys
Parameters
No parameters


GET
/prop/byagent/{id} Properties filtered by Agent
Parameters
Name
Description
id * required
integer($int64)
(path)
id
Responses
Code
Description
200
Found the Property
Media type
application/json
Controls Accept header.


GET
/prop/bb/{mm}/{id}
Properties filtered by number of bathrooms and Bedrooms
Parameters
Name
Description
mm * required
integer($int32)
(path)
mm
id * required
number($float)
(path)
id



Example return:



2 Self-Evaluation. Critically appraise the strengths and weakness of the API you have developed.

While creating the REST API for this assignment I uncovered some blatant strengths and weaknesses along the way. Below are a few of the main pros and cons I have discovered.

A strength of my API would be the exception handling. I have implemented what I believe to be a well-structured exception handler that clearly states what the exception is. There is nothing more infuriating in development that encountering an error and not knowing how or where to solve the problem. Through the use of exception handling, I believe the API clearly states why an exception or error has occurred. To complete this I created three classes within my system which all played a part in dealing with the different types of exceptions to the system. Depending on the exception a unique error message would be displayed along with the current time and Http Status.

Another strength of my API would be the variety of API I have created. When creating API, I believe the more you can do with it the better standard it is. If for example all my API could so was delete, update and add it would be very basic and would be very easy to implement. However, I believe the API is of adequate standard as I have developed functionality such as a QR Code, PDF documentation creation and also a function that returns a zipped file. The API to have the ability to automatically create a PDF file with related information on a given property. This is a feature that I think would be of particular use in the real world. As it would be available at the click of a button it could come in very handy in any realtor's office.

One major weakness of the REST API is the inability to perform more than one operation at a time. For example, if I wanted to insert fifty new records into the properties table, I would have to perform exactly fifty identical operations. This would be very time consuming and monotonous. It would not be a very practical approach and should be avoided if at all possible. I think that it would be of benefit to me in the future to research into REST API that could carry out numerous operations at the same time.

Another weakness of my API would have to be security. There is no proper protection of the data and from a security standpoint I would have to say my API is vulnerable. I have not implemented any special security to the system. I wouldn't be an expert on hacking or software attacks, but I do not believe it would be difficult for a hacker to maliciously effect the data in the database or conduct something of similar negative impact.

Overall, I believe I have learned about the general pros and cons when it comes to the creation of API. I think this assignment has helped me understand the potential uses of REST API and in what scenarios the development of such API would be useful.

3 Benchmarking. Measure the benefits that the development of an API brings to a development project.

The development of an API can bring numerous benefits to any development project. One of the main benefits an API can bring is the convenience of automation. Through the use of an API, computers and machines can be in charger of different aspects of a system or company, removing the possibility of human error and also expediating the process.

This follows on to another benefit which is efficiency. With the use of an API content can be generated automatically very swiftly. API support allows developers to reach their deadlines and achieve more with less time. This can have a massive impact on a development project as it frees up extra time for the developers allowing them to focus their efforts on different parts of the project.

Another benefit is innovation. The creation of APIs can lead to alternative and innovative methods which can make a project stand out and perform better than similar systems. There is lots of techniques that can be applied when creating API which occasionally may lead to outside the box ideas.

No development project wants to fail. Through the development of APIs developers can decrease their risk of failure. A system that is unsustainable and provides contradictory customer experiences is doomed to fail. This is a situation that is avoidable with an API first approach. This isn't a concern for teams who are focused on creating APIs first. Throughout any development lifecycle user requirements may change or it may be necessary to alter some of the system. This is an area that is not a major problem for an API. APIs are very scalable and relatively easy to alter based on what a user requires. An API-first strategy's natural adaptability that allows for rapid modification to evolving requirements and more comprehensive involvement of end-users in the development process.