Rentivo/Lycan Integration Design Doc (Listings)

Date: 10 Jan 2019
Author: @kcvan
Approvers: @tommychheng @jeremykwa @tonytran @woeltjen

What is this feature?

Integrate Rentivo listings into Beenest

Who will use this?

Guests

Implementation Overview

`Current/New dbee listings API/service functions/new constuctors Lycan API's`

New flow with Rentivo integration:

1) Guest searches for a listing from search bar -
    a) Client-side calls `searchListings` gql resolver
    b) Gql resolver calls `ListingService.searchListings`
        i) `ListingService` uses a `listingAggregator` constructor -
            (1) `listingAggregator` is a new constructor that takes in listing providers eg. `rentivoListingProvider`, `beenestListingProvider`, as parameters and holds methods pertaining to listings. This can include merging listing providers, mapping provider listing properties to our listing properties, etc.
                (a) `rentivoListingProvider` will be a new class that implements `Lycan APIs` (more specific API calls shown in implementation details) that have to do with listings
                (b) `beenestListingProvider` will be a new file that we move the business logic of searching listings to.
    c) `ListingSevice.searchListings` will then use: `listingAggregator.search` and return an array of merged listings

        i) `listingAggregator` will have a method that would merge provider listings
2) Guest clicks on a single listing -
    a) Client-side calls `listing` gql resolver
    b) Gql resolver calls `ListingService.getListingById` with the Rentivo listing id
    c) `ListingSevice.getListingById` will then use `listingAggregator.getListingById` and return a formatted listing (whether it's from Beenest, Rentivo, etc.)
        i) `listingAggregator` will have a method that would pull single listings from providers and match their listing properties to our required listing properties
3) Guest clicks on booking card calendar -
    a) Client-side calls `reservations` gql resolver
    b) Gql resolver calls `ListingService.getReservations`

In order for the 3rd scenerio to happen, we will need to get iCal links from Rentivo (they have iCal links) and update our reservations table with the reservations/block-out dates from Rentivo.

List the security concerns and attack vectors

How is this an improvement over the existing system?

We can add many additional listings through Rentivo without much work on the product side. The influx of supply will hopefully spark more interest and bookings on Beenest.

Implementation Details

Backend

Lycan APIs and their purpose:

```
Lycan Listings: POST - Get Authentication JWT (Bearer)
@params {string} username
@params {string} password
@params {string} application_hash
```

curl --location --request POST "https://lycan.rentivo.com/api/oauth/token"

--header "Accept: application/json"

--header "Content-Type: application/x-www-form-urlencoded"

--header "Authorization: "

--data "_username=&_password=&application_hash="

This endpoint is used to get data from Lycan's APIs. Needed on all requests that are not public endpoints

Lycan Listings: GET - Fast Realtime Pricing/Availability

@params {string} listingId
@params {string} arrival
@params {string} currency
@params {string} departure
@params {string} guests

Lycan endpoint with dummy data:

curl --location --request GET
https://lycan.rentivo.com/api/public/listing/{{listingId}}/pricing?arrival=2018-11-10&currency=EUR
&departure=2018-11-17&guests=1

--header "Accept: application/json"
--header "Content-Type: application/json"

This endpoint can be used to:

- Find out if a property is available
- Find out if a property has pricing (just because a property is available does not mean that the host/owner has configured pricing or that the property is actually bookable)
- Find out if the property is bookable. For a property to be bookable it must not have any errors, and be Available and have Pricing.
- Get terms and conditions for the listing
- Discover if the listing has a payment gateway attached for instance bookings (by looking at the pairings array, which is the gateway pairings)

Lycan Channels: GET - List (might be replaced with elastic search)

@params {string} channelId
@params {string} page
@params {string} count

@params {‘schemaObject’, ‘memberMappings’, ‘provider’, ‘channel’} expand

Lycan endpoint with dummy data:

curl --location --request GET

https://lycan.rentivo.com/api/channels/{{channelId}}/listings?page={{page}}&count={{count}}&expand=schemaObject

--header "Accept: application/json"

--header "Content-Type: application/json"

--header "Authorization: ResourceKey {{token}}"

This endpoint is used to get all listings associated with a channelId

`Lycan Channels: GET - Show`

@params {string} channelId
@params {string} listingId
@params {‘schemaObject’} expand

Lycan endpoint with dummy data:

curl --location --request GET

https://lycan.rentivo.com/api/channels/{{channelId}}/listings/{{listingId}}?expand=schemaObject

--header "Accept: application/json"

--header "Content-Type: application/json"

--header "Authorization: ResourceKey {{token}}"

This endpoint is used to get the data related to a single listing

Tasks:

1) Create Providers
    a) Create rentivoListingProvider
        i) Implement Rentivo APIs in this file including:
            (1) `Lycan Channels: GET - Show`

(2) `Lycan Channels: GET - List (might be replaced with elastic search)`

b) Create beenestListingProvider

i) Move searchListings business logic from ListingService to beenestListingProvider

2) Create listingAggregator constructor that accepts rentivoListingProvider and beenestListingProvider as parameters

a) Implement searchListings method that accepts and merges multiple provider listings into one and returns an array of meged listings

b) Implement convertRentivoListing method that accepts and matches Rentivo listing properties to our listing properties and returns a listing object

c) Edit ListingService.searchListing to use listingAggregator.searchListings

d) Add getLycanToken

3) Update iCal function that adds iCals to our iCal table to take Rentivo iCals

Property matching

Bee Listing Properties	Rentivo Listing Properties
`addressLine1`	address: addressLine1
`addressLine2`	address: addressLine2
`amenities`	Features: { category: amenities } (array of objects that have the category of 'amenities'
`checkInTime`	arrival: checkInStartTime arrival: checkInEndTime
`checkOutTime`	departure: checkOutTime
`city`	address: city
`country`	address: countryISO2 (need conversion)
`description`	descriptiveName
`homeType`	listing: type
`houseRules`	description: en: content (look for description: en: title: 'What are your policies'. Note: this

	might be different depending on host, all of the content have type: ‘FAQ’ so might need to just search and show all of these)
icalUrls	RENTIVO TO PROVIDE
isActive	flag: isActive
lat	address: latitude
lng	address: longitude
listingPicUrl	media: anyOf: title: PHOTO && Media: anyOf: isFeatured
maxGuests	listing: maxOccupancy
minimumNights	pricing: dynamic: data.findObjectInArrayWith(minimumNights)
numberOfBeds	rooms.findObjectInArrayWith(type: BEDROOM)
numberOfBathrooms	rooms.findObjectInArrayWith(type: BATHROOM)
photos	media.findObjectInArrayWith(category: PHOTO)
postalCode	address: zipPostalCode
pricePerNightUsd	How do we deal with variable pricing?
securityDepositUsd	??
sharedBathroom	Never shared
sleepingArrangement	??
state	address: stateProvince
title	headline: en: content Else descriptiveName

Frontend

The front end should be the same for guests. The main difference is that the host will not go through our host portal, but Rentivo's.

Maintenance/Support Details

Backend

1) What happens if a service is down? How would we handle this?
2) How do we sort the listings that are merged together? Is there any priority?
3) How do we deal with the situation where a service takes too long to fetch data from a source?
4) How should we deal with listings that require days booking in advance?
5) How do we deal with dynamic deposits? These are virtual properties that we can't account for and being able to list out the different fees/deposits that listings have would be good.

Frontend

N/A