AWAD – Assignment Two

Individual Project

Konstantin Milchev

**Contents**

**Application Requirements – p3**

**Vision – p3**

**Application Features – p5**

**Non-Implemented Features – p8**

**Challenges – p9**

**Appendix – p10**

**Application Requirements**

* A terminal to run ruby code (unless using Heroku)
* Gemfile – Bundle install within terminal to make sure all required gems are installed.

**Version Control**

The application was initially controlled by using bitbucket, the repository was successfully cloned onto GitHub.

* git@[bitbucket.org:](http://bitbucket.org/)art3xias/werent.git
* https://bitbucket.org/art3xias/werent

**Administrator Credentials**

Username: kircho@abv.bg

Password: qqqqqq

**Vision**

The purpose of the application is to provide people looking for a place to rent with a modern, simple and easy to use website. The search can be done through name, number of bedrooms, bathrooms, whether there is internet or heating and via available dates.

**Basic Application Features**

The search function which is diverse and accurate and the calendar datepicker.

**Custom CSS Style Sheet**

We implemented a stylish custom CSSS sheet to make the application more attractive to users. With many users viewing web content on mobile devices, we also decided to implement Bootstrap and toastr-rails[[1]](#footnote-2) to make our web application responsive and suitable for the range of screen ratios available. Bootstrap also ships with the powerful Sass pre-processor which made it simpler to create the graphical interface.

**User sign up + Profile Page**

To allow users to create their own content on the website we created an HTML form which allows users to sign up and create profiles. This feature allows users to generate content through the become a host function and also leave reviews, share messages with one another and personalise their profiles with a profile picture. Users wishing to upload a profile are required to sign up for a gravatar account.

We implemented validations to ensure name and email(user) and home\_type, number of bedrooms,number of bathrooms,summary, picture, etc (room) attributes couldn’t be left blank and because the user email address is required to be unique, validations also prevents multiple accounts from being set up for one email account. The application also checks if the e-mails provided by the users are in a legitimate format.

To make sure our user accounts are secure we also used validation tests to ensure users pick a password of at least 6 characters, furthermore, to keep our users passwords secure we decided not to store passwords on the database but instead a hashed version of a user’s password. If our database is compromised user passwords will not be compromised.

**Authentication System**

To further enhance user experience when using the WeRent app, I used the devise gem for Authentication.[[2]](#footnote-3) Using it made developing the users much easier and quicker by creating a user model for regular use, admin migration for the admin role and using the before\_action method to disallow access to certain pages for the guests.

Most importantly, devise allows us to control the features each user has access to when using the web application. For example, administrators are able to edit and delete accounts/reviews where as a regular user are only be able to add and delete their own accounts/reviews. Furthermore, once a user is authenticated they will be able to access their own profile, view rooms which they have made and update profile settings. In addition, using devise , a useful “remember me” feature was added to the login screen, making it easier and quicker for regular users to log in. Devise also allows the use of ‘timeout\_in’ method, which will log the user automatically after several minutes of inactivity. Unfortunatelly, only one of the two(remember\_me and timeout\_in) methods could be used at the same time so I decided to go with the remember me method.

**Account Activation + Password Reset**

Another feature we implemented to help verify users is that of account activation. Like the authentication system, whereby a token was used to verify a user, tokens are also used as part of the account activation procedure. An activation token is generated during account creation then signed and sent to the user e-mail address. The account isn’t activated until the user clicks the link within the email confirming the activation token. The activation token is returned decrypted and it’s hash value is compared to the token digest stored in the database. This is a security feature that reduces the chances of bots creating fake accounts on our databases with the purpose to cause system failure. It also protects from attackers trying to gain access to our database or steal the user’s cookies. Nevertheless, there were complications with this. More can be found in the challenges section.

If a user forgets their password, a reset token can be sent to the users email address (same process as the activation token) with a link to reset their password.

**Admin Options**

Implementation of admin controls was necessary to control content that is uploaded to the website as well as to maintain the database. Administrators can edit and delete all reviews and user accounts. This differs from regular users whom are only able to update their own information. While the authentication system allows us to identify who system users are, it is the authorisation system that allows us to control what features they access. Having built a devise authentication system, it was straight forward to implement our own authorisation system as well. If a user tries to access a page they are not authorised to view they are redirected to the home page.

**Reviews**

Added a review option in each room. Jquety was used in order to calculate the average rating and total number of people who have reviewed a room. Also star images were used in order to display the average rating.

**Messages**

Messages and Conversations were implemented between users. Through a sender\_id and recipient\_id the messages are being transferred between one another. ‘private\_pub’[[3]](#footnote-4) gem was implemented for the users to have live conversations where the first user sends a message and the second user instantly receives it, without needing to refresh the page.

**Facebook Authentication**

This was completed through the gems ‘omniauth’ and ‘omniauth-facebook’. An api key is generated through the developers section of the facebook website and this allowes a user who does not want to register to log in with their facebook account.

**Search Function**

The search function was implemented via the ransack gem, which generates simple forms(form\_for turned into seach\_form\_for). The query includes all of the attributes of the Room model from home\_type to is\_pets(if pets are allowed).

**DatePicker**

The datepicker required jquery in order to generate live calendar data and also save the dates which have already been booked. The gem ‘jquery-ui-sass-rails’[[4]](#footnote-5) was used.

**Challenges**

1. **Account Activation**

I could not register for SendGrid, MailChimp and Mandrill. I was flagged for a high risk account and when I contacted them saying I was a student needing the service for a project they said that their algorithm makes the choices for them so there is nothing they can do. I tried with several accounts nothing worked. The application still sends the activation token to the terminal but without an additional email. Thus, the only feature lacking is the email activation.

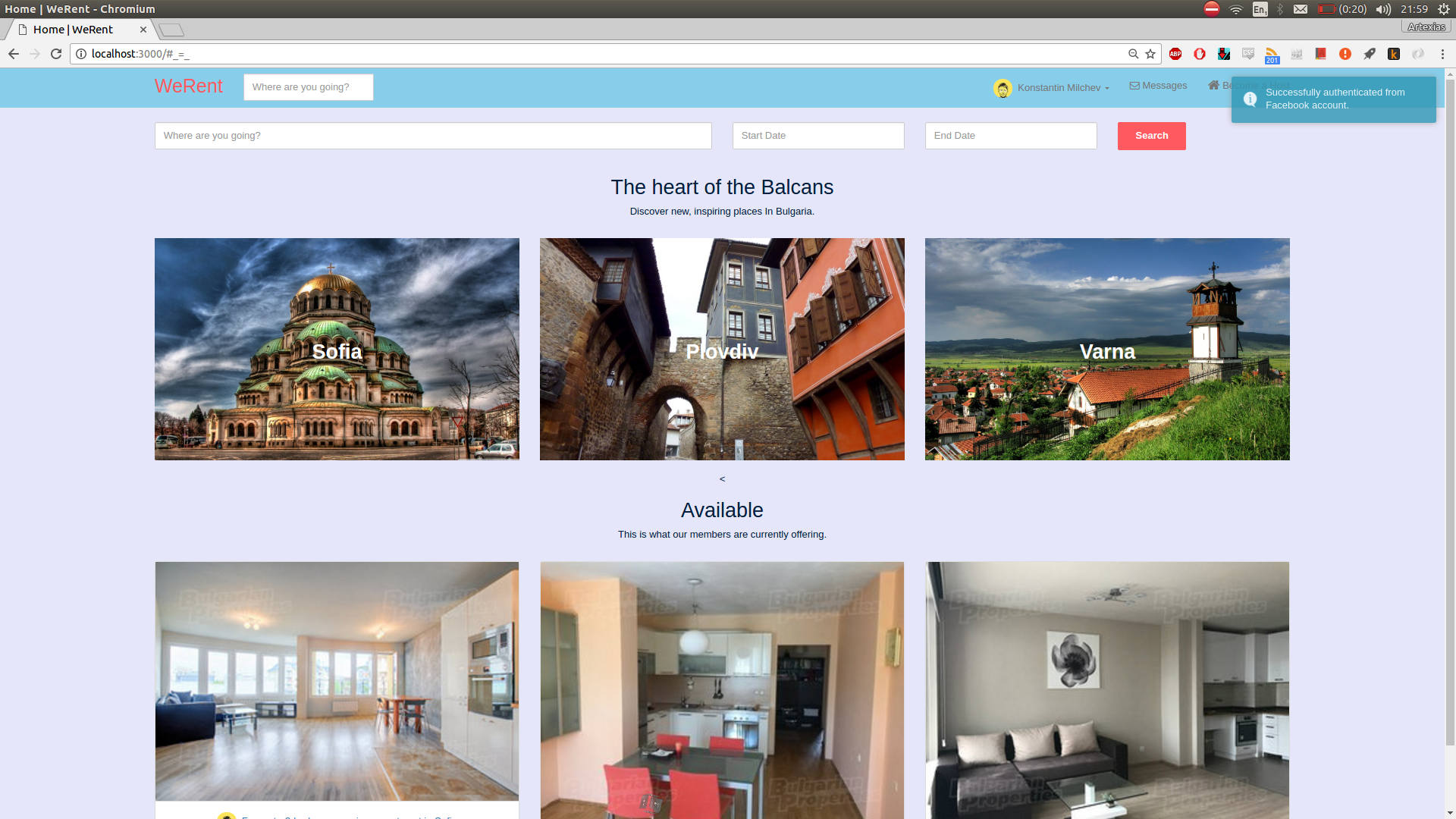
2. **Video Background**

I tried to include a video background for the search are in the home screen something like this website.[[5]](#footnote-6) But I could not get it to work, most of the time it kept freezing my search function, thus due to time limitations I could not implement it.

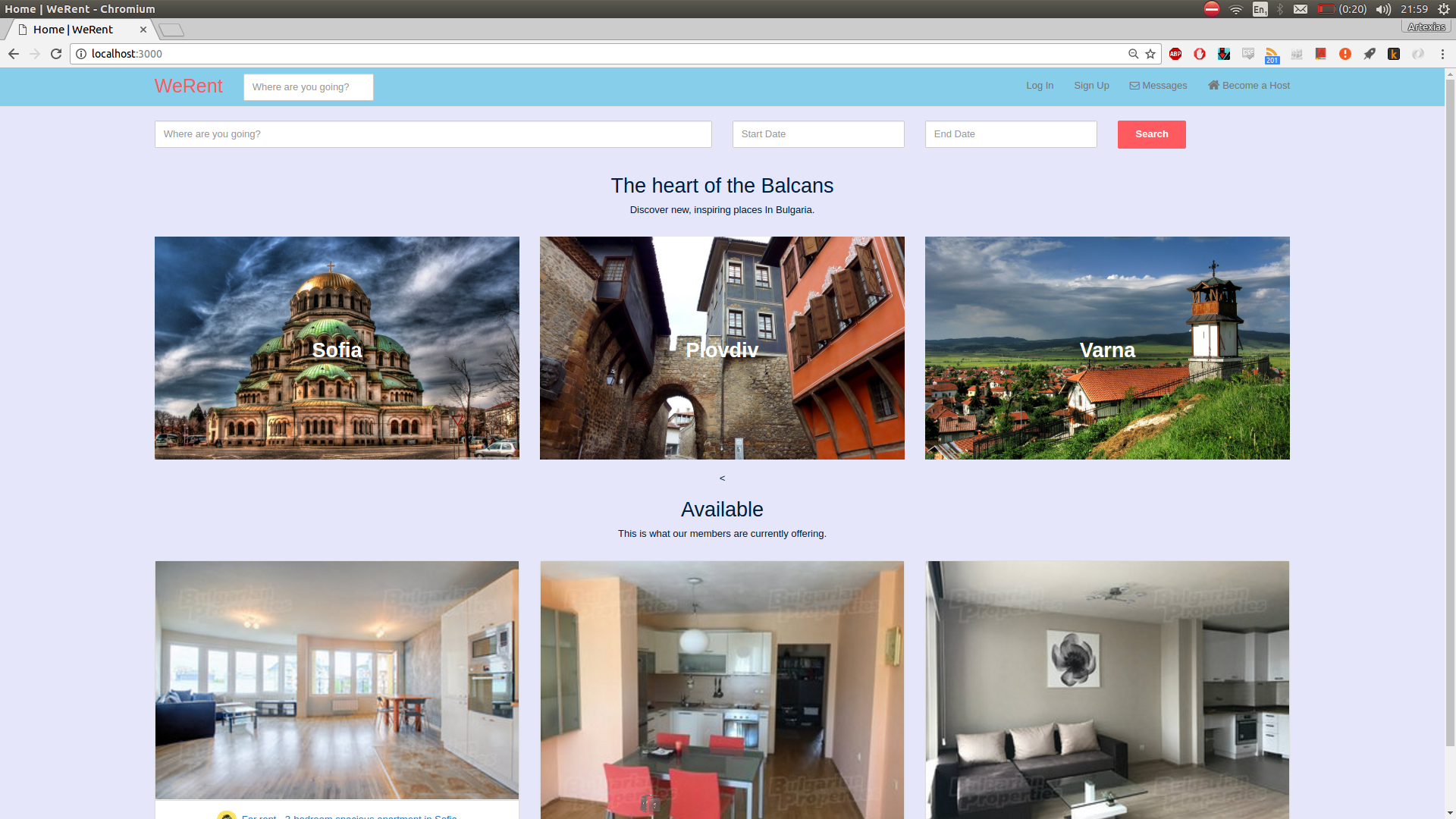
**Appendix**

**1.** Log in Page

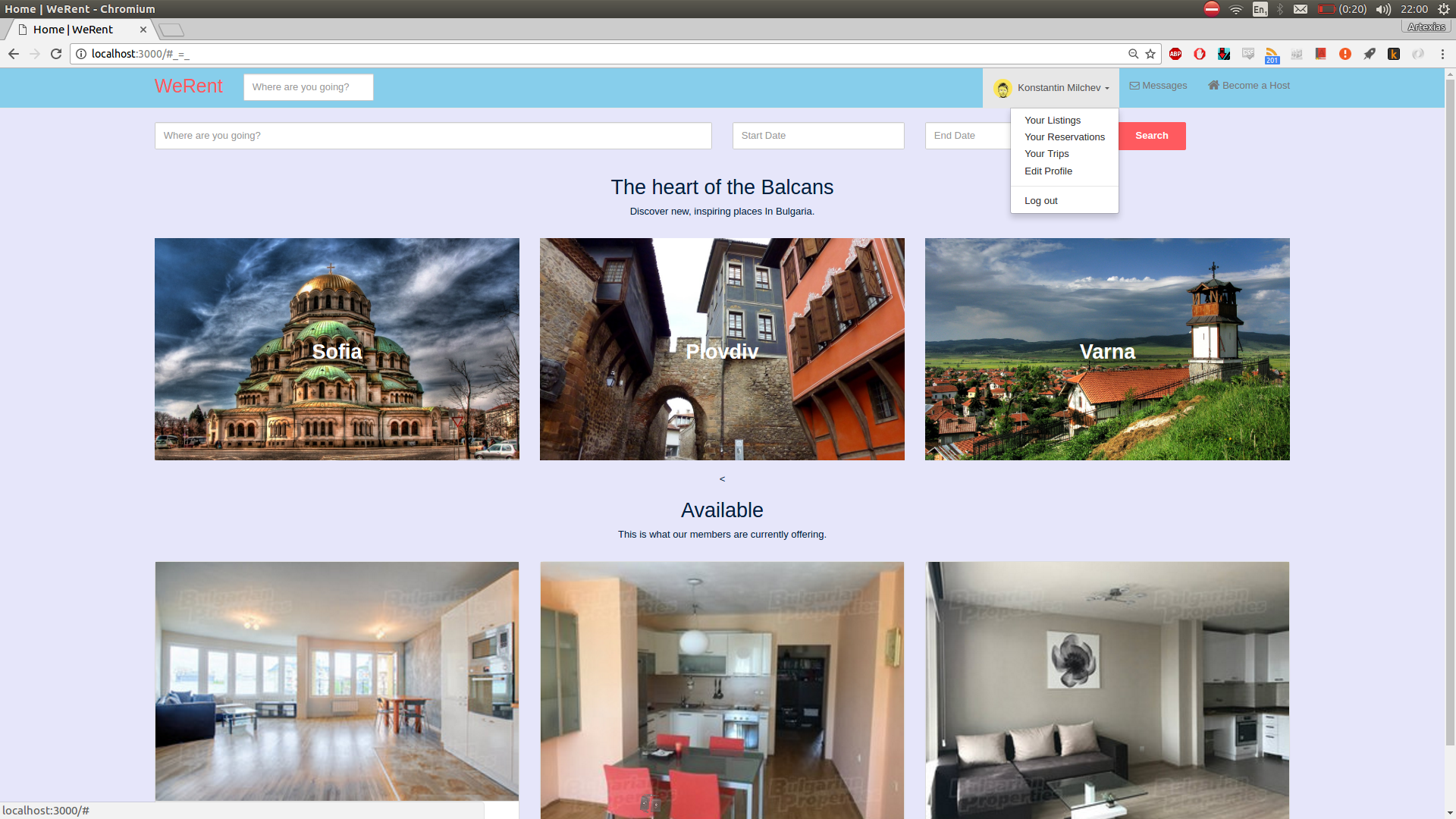
**2.** Facebook logged in

****

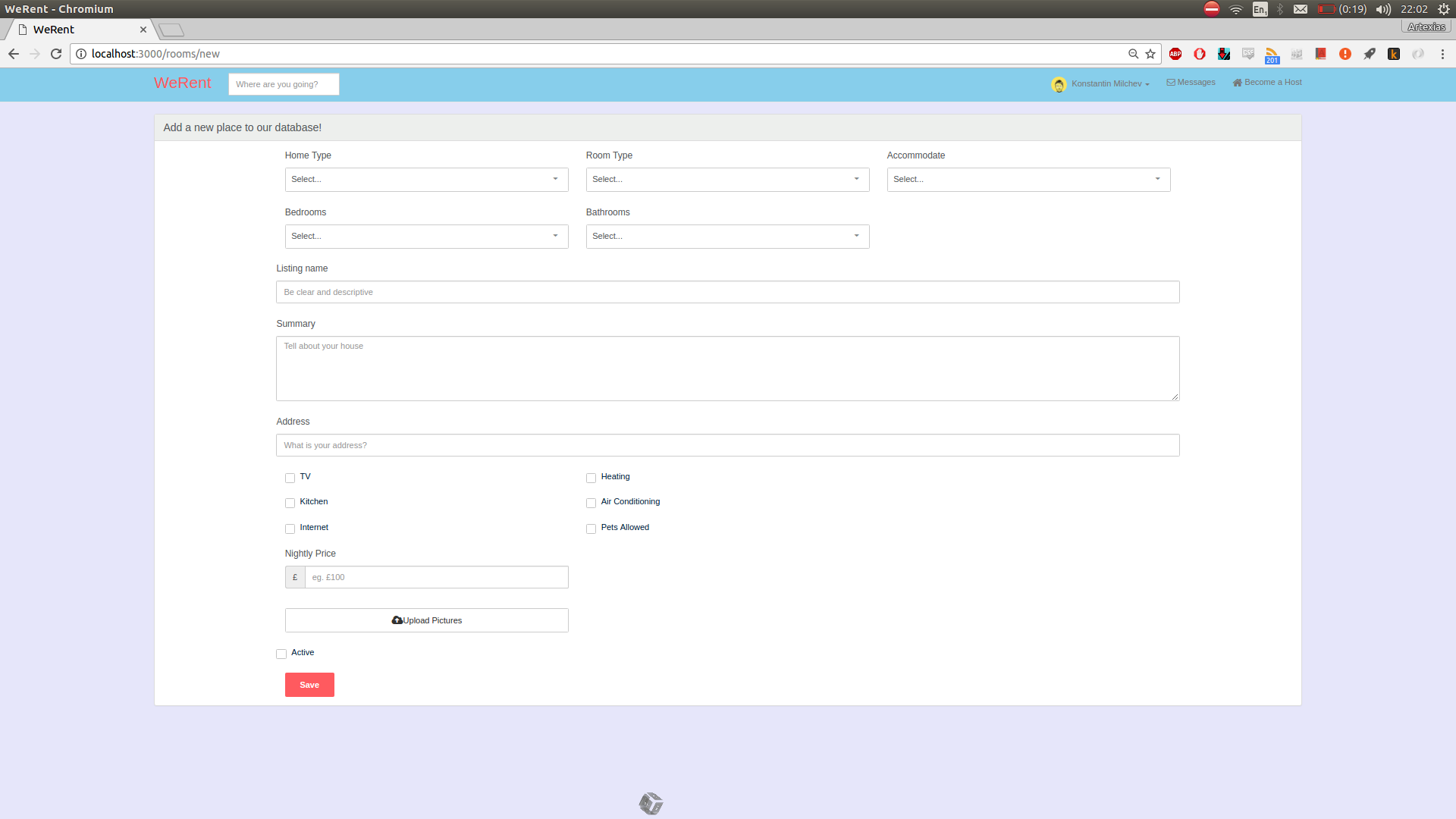
**3.** Main Page

****

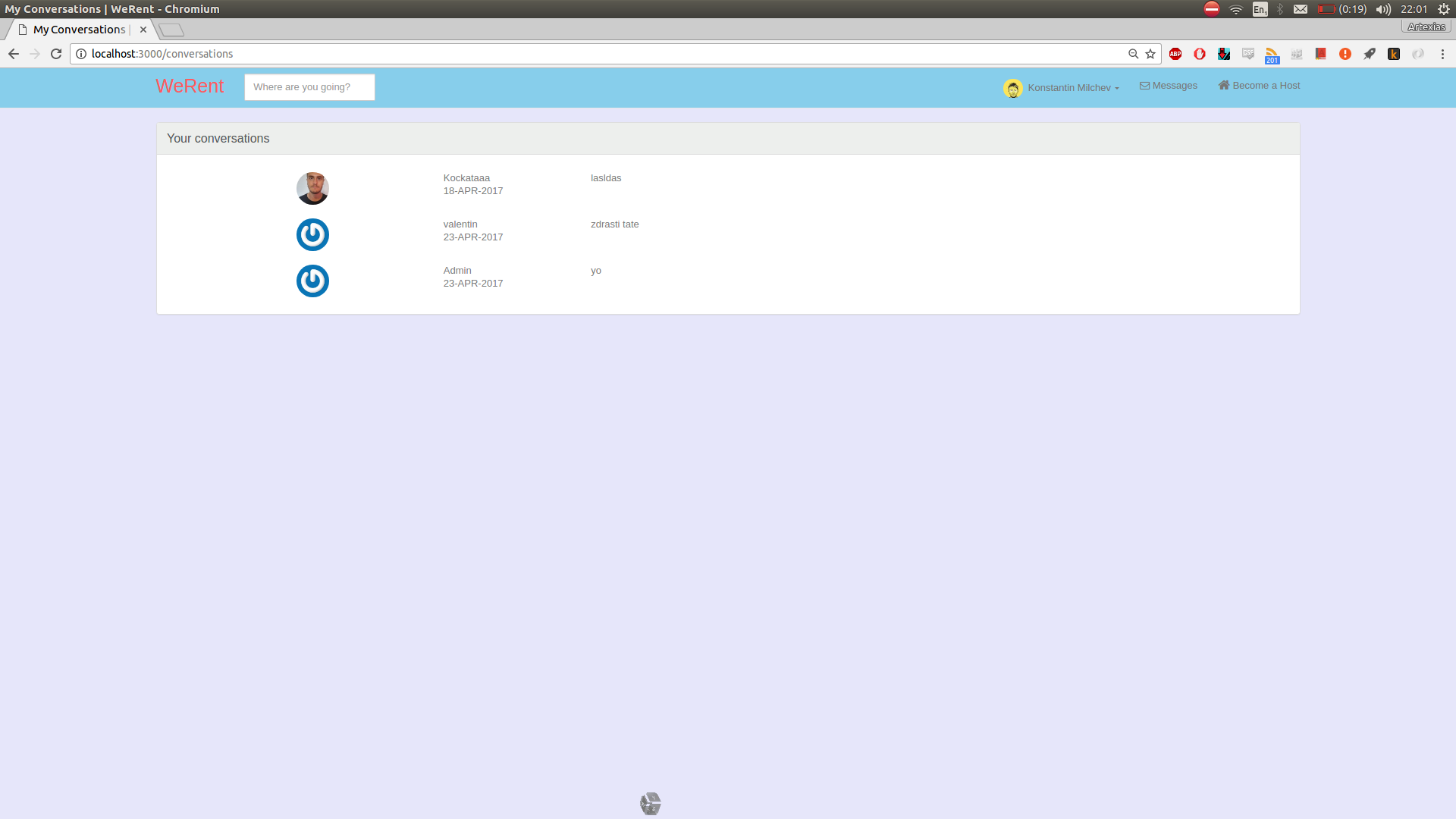
**4.** User Options

****

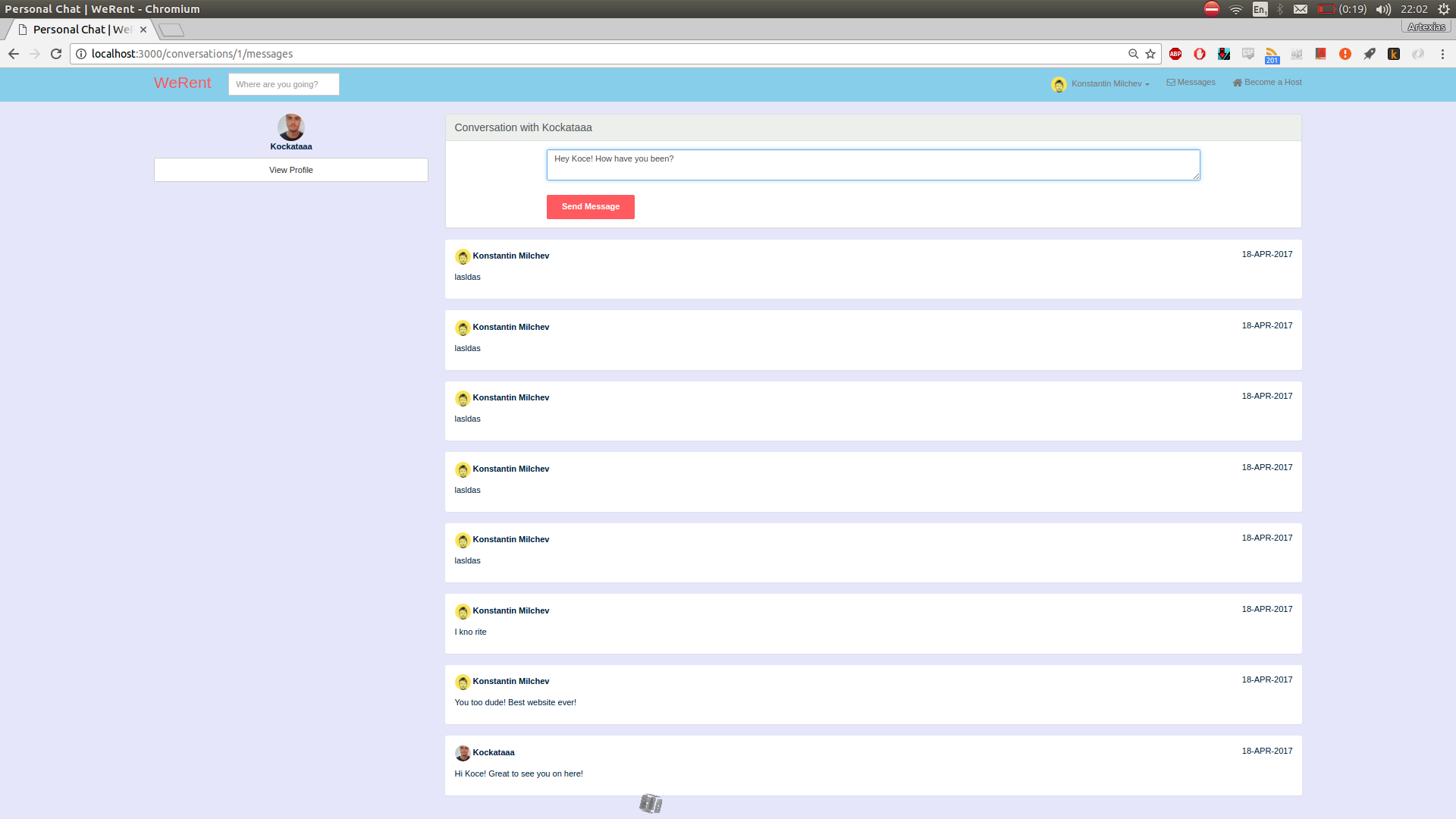
**5.** Become a host

****

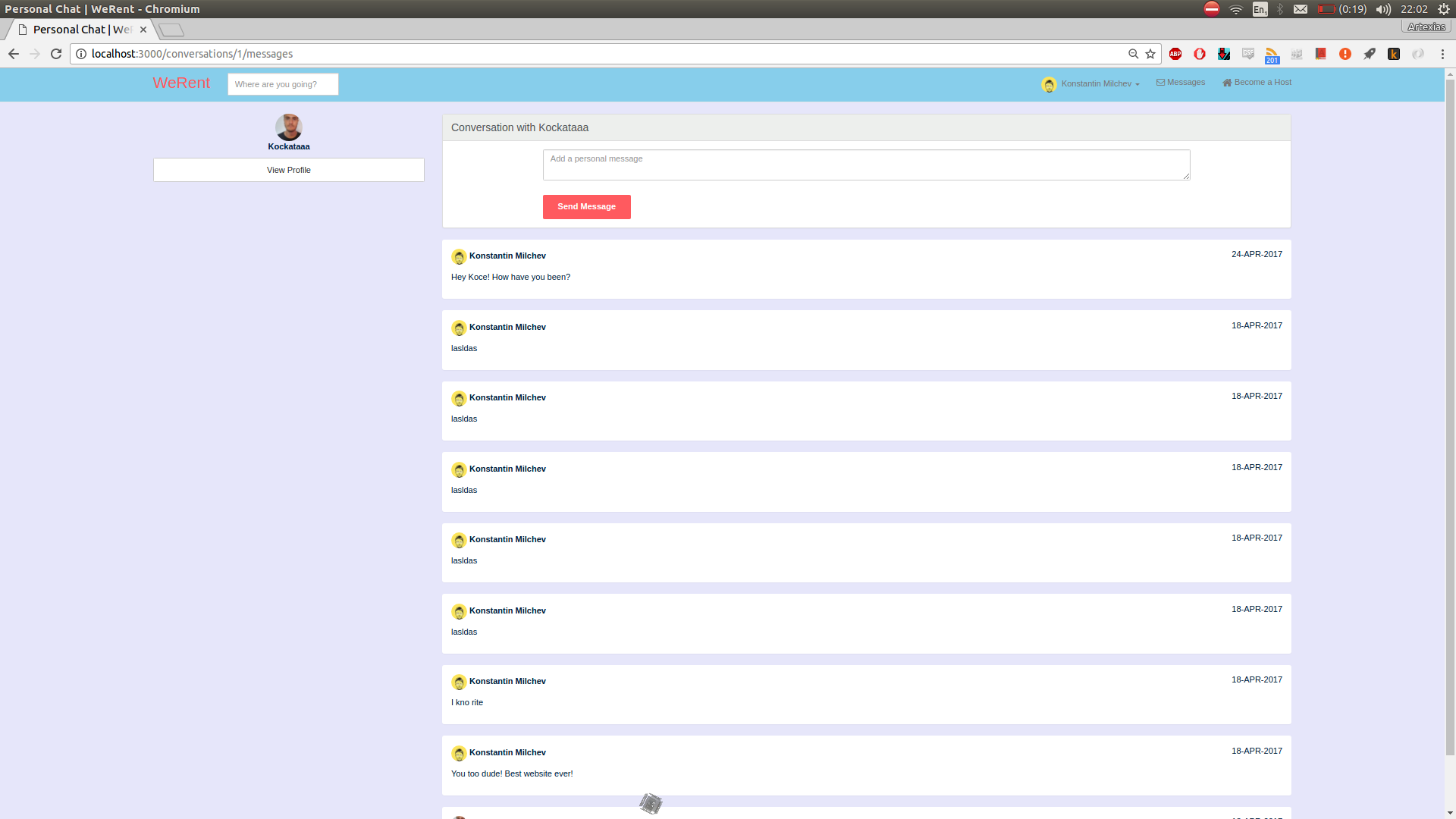
**6.** Conversations

****

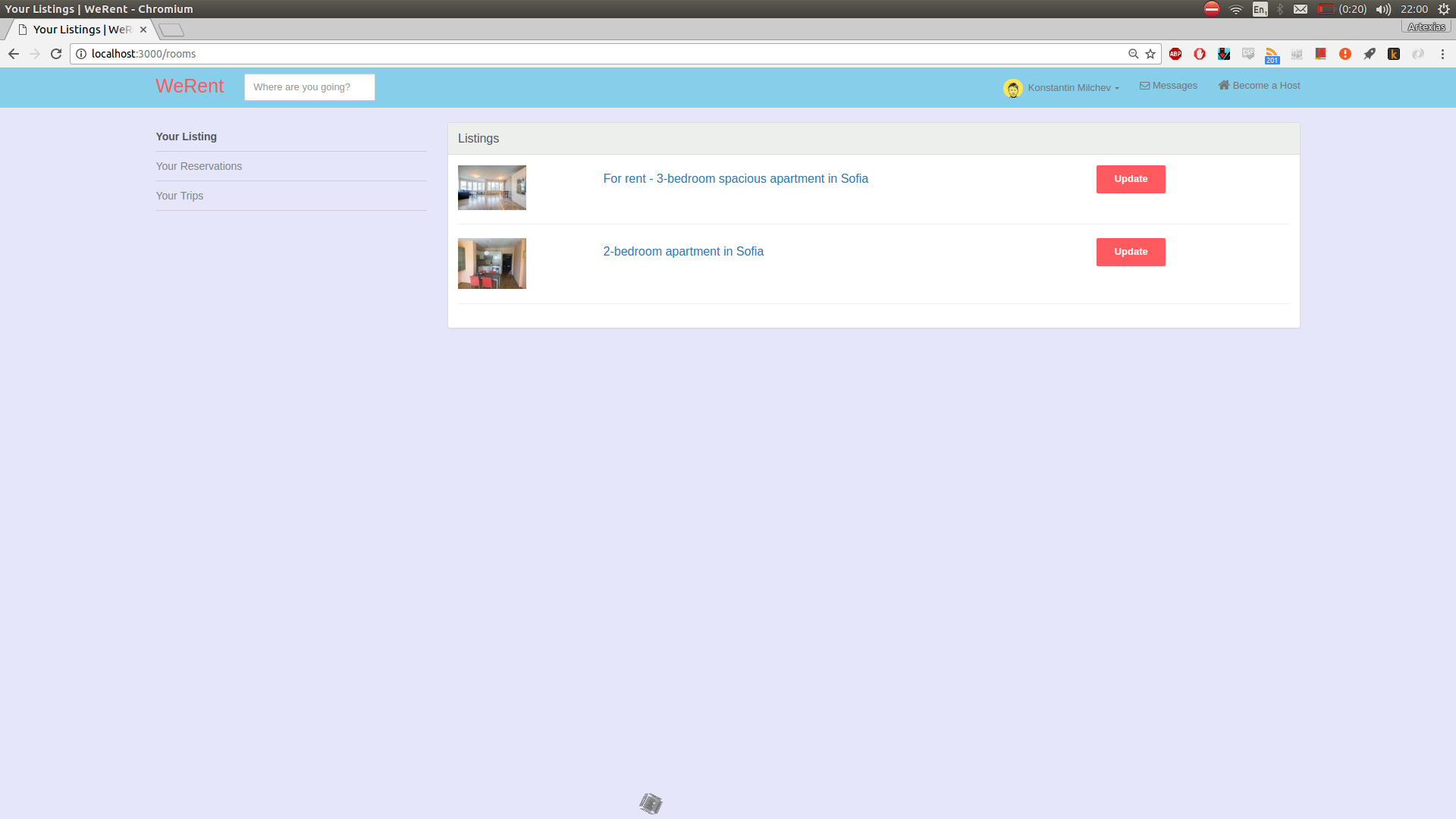
**7.** Messages - 1

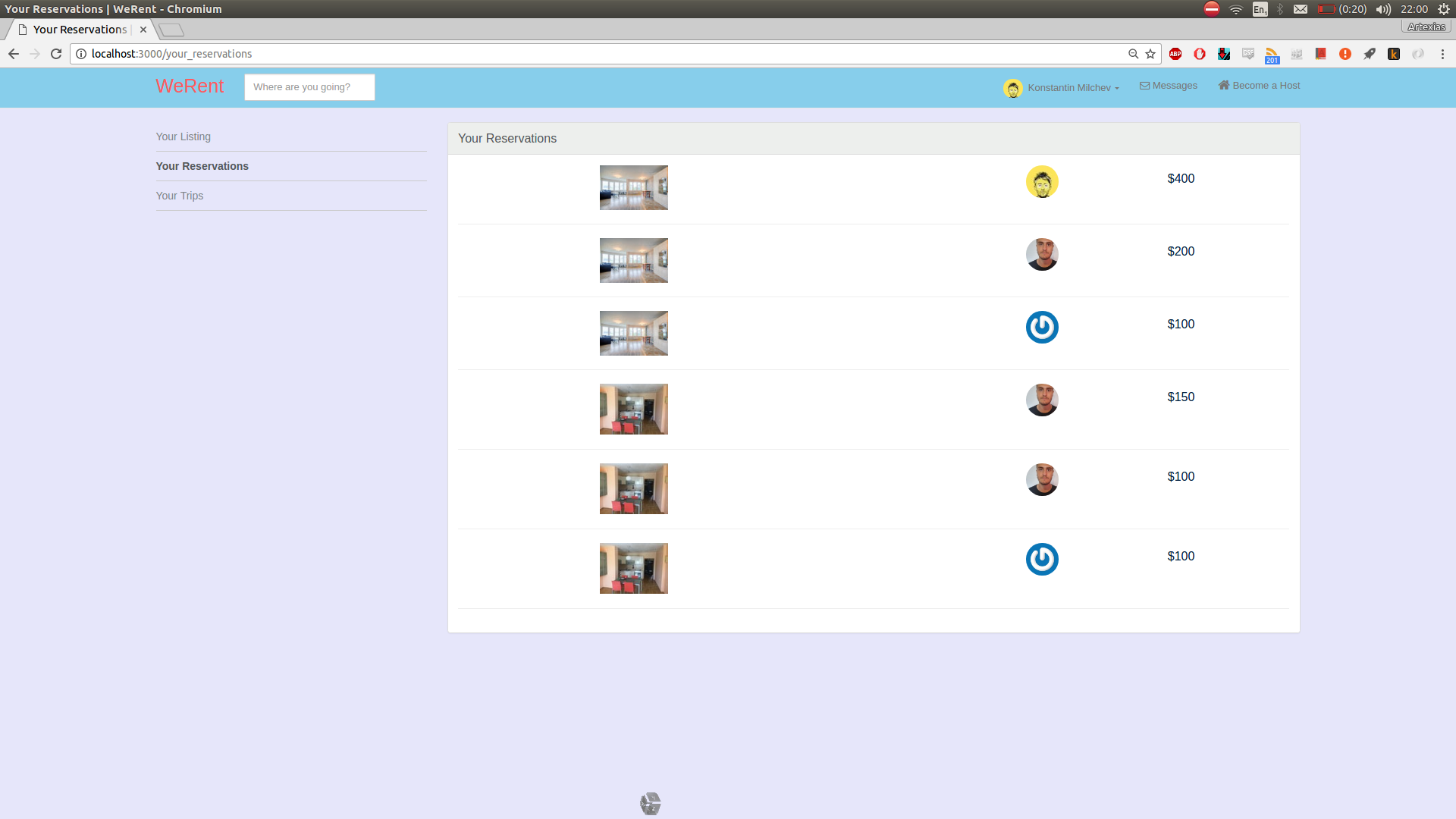
****

**7.** Messages – 2

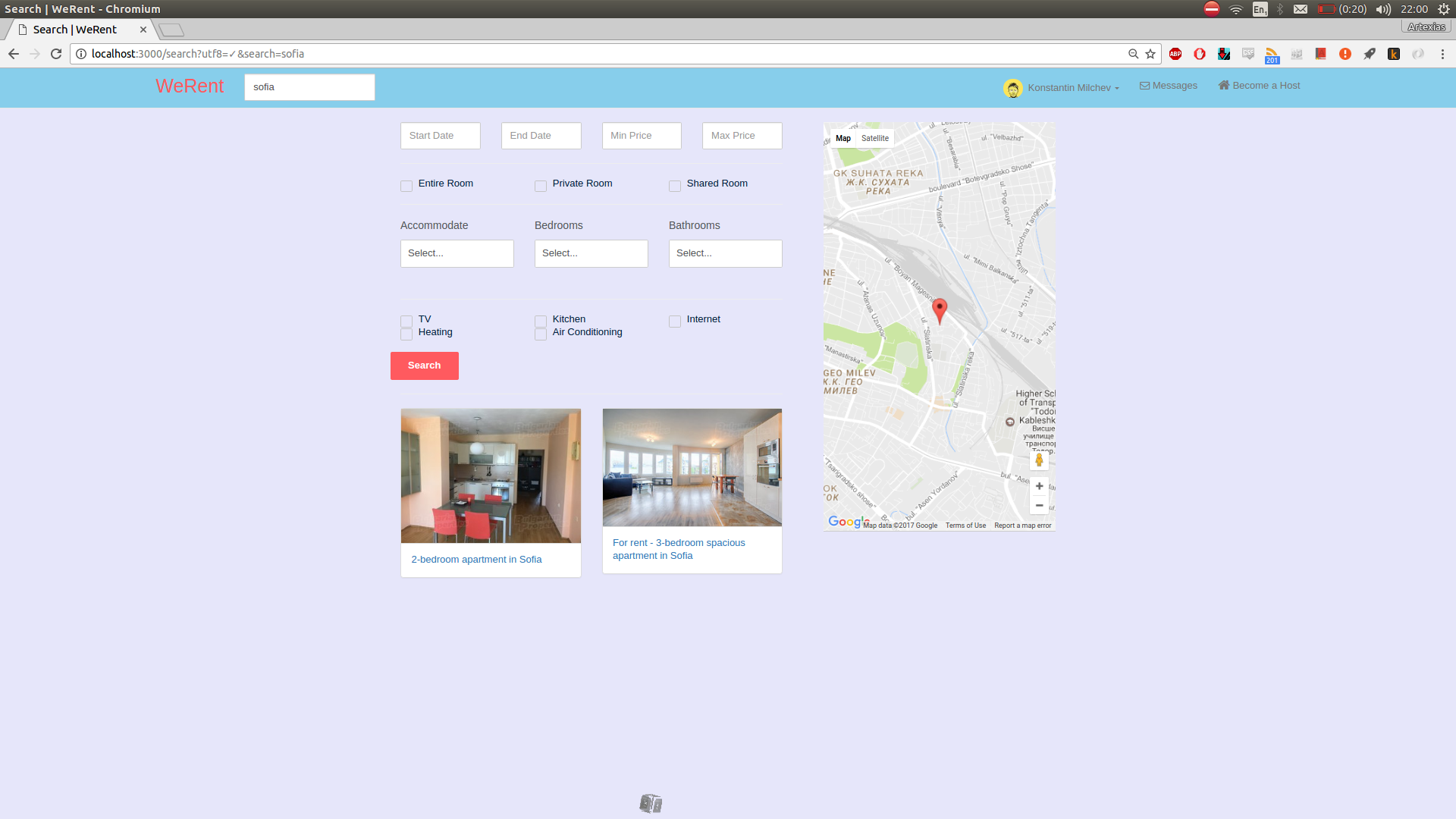
****

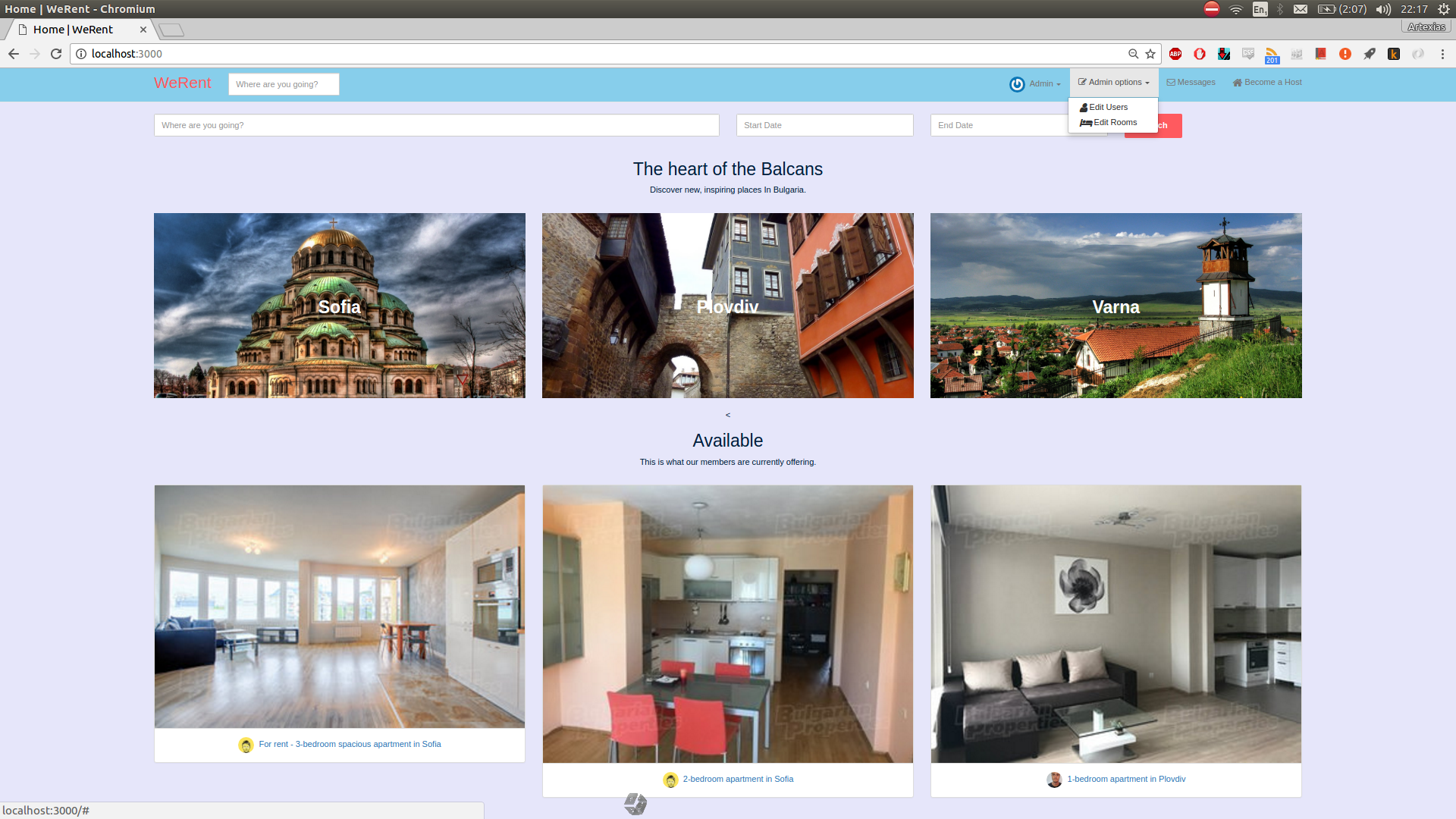
**8.** Listings

****

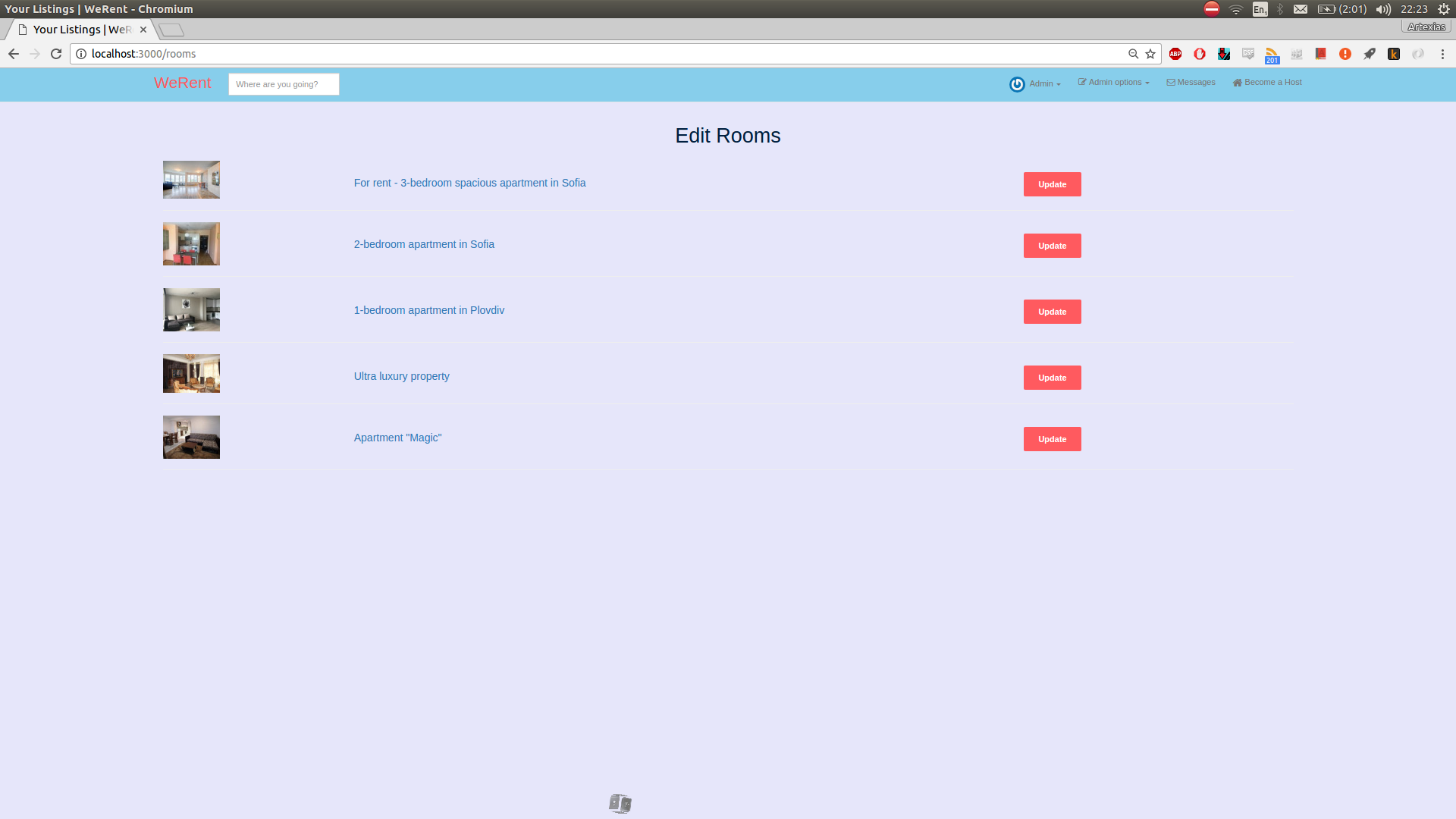
**9. Reservations**

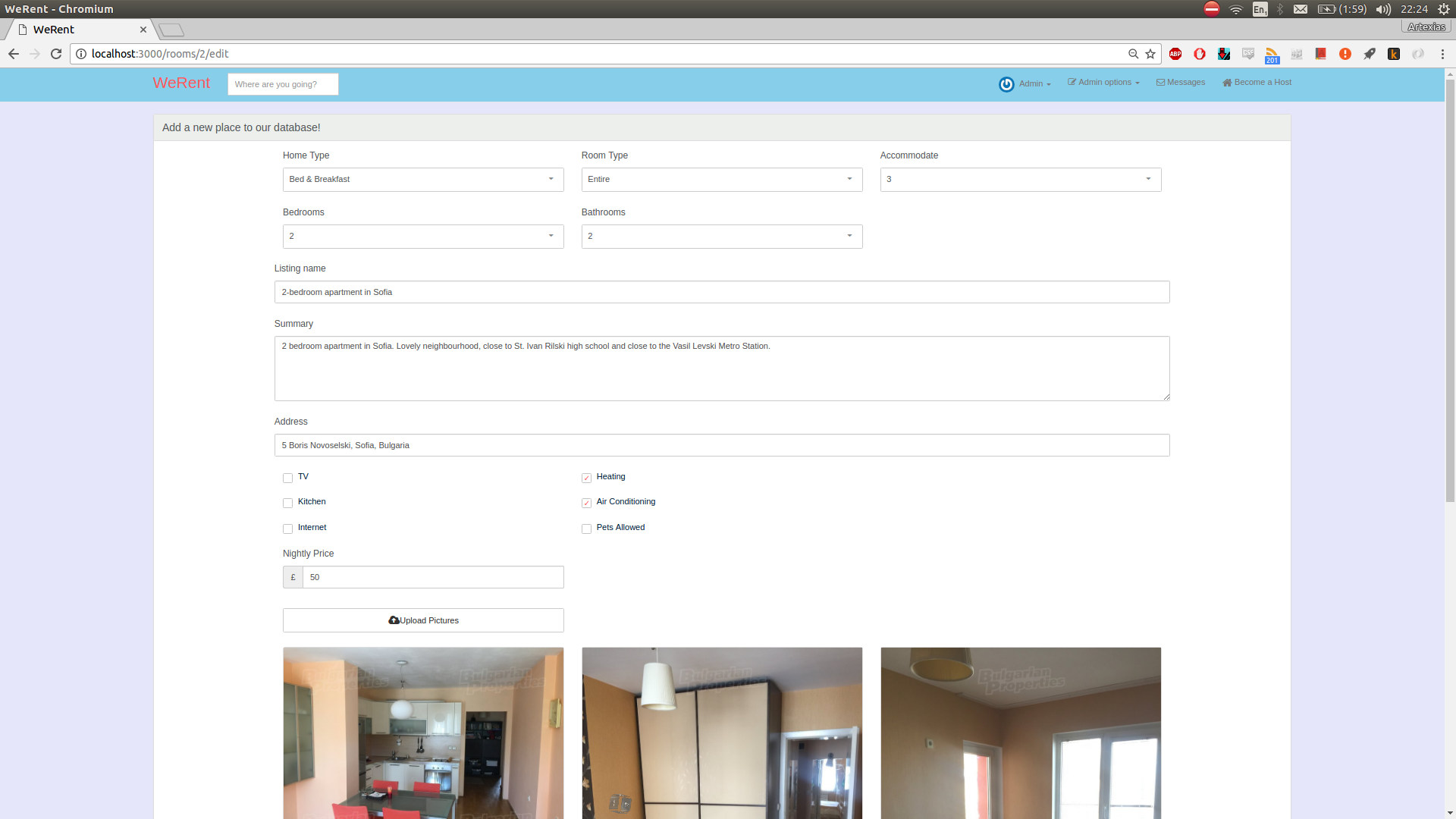
**10. Search for “Sofia”**

****

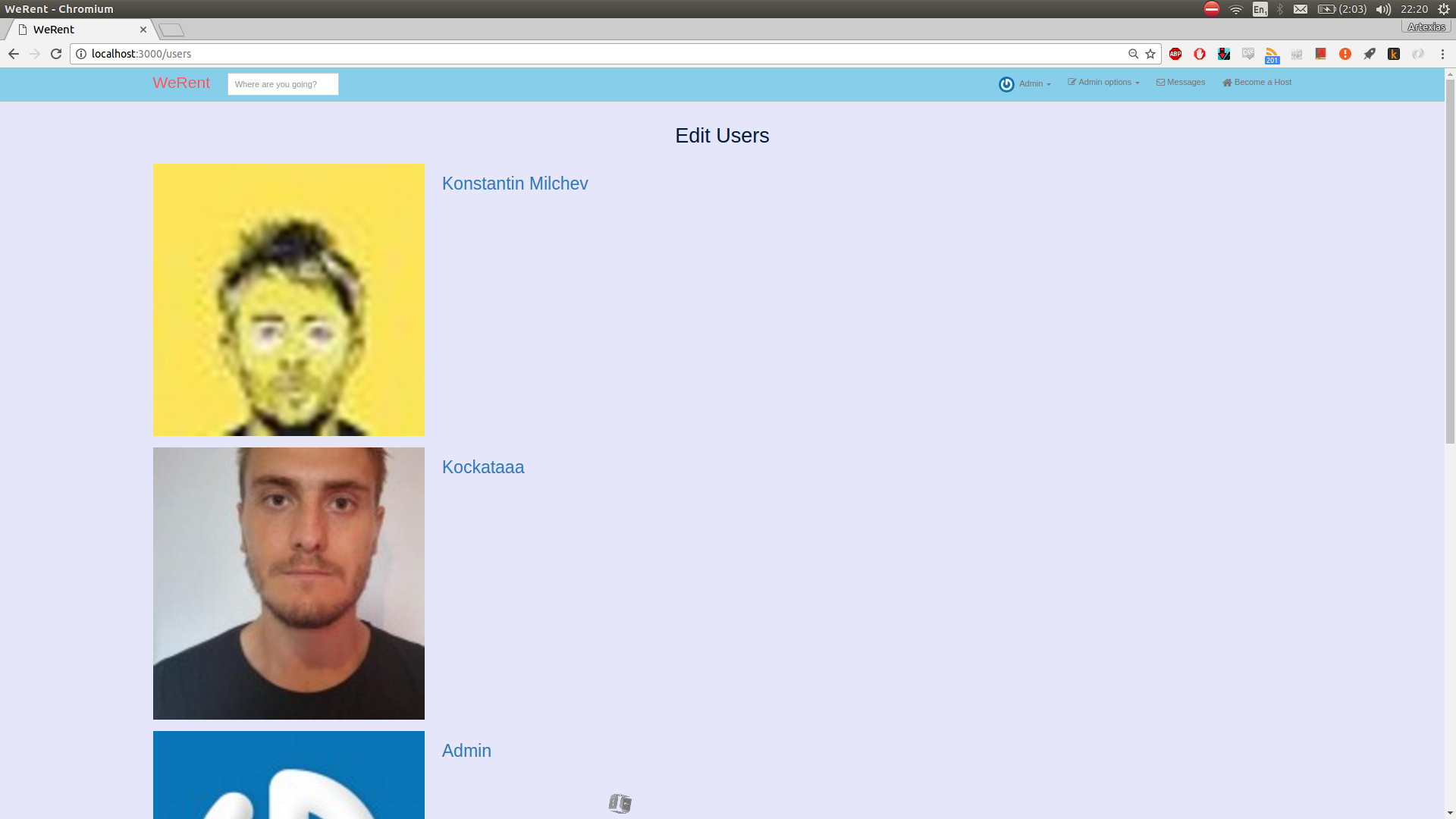
**11.** Admin Profile

**12.** Edit Rooms – 1

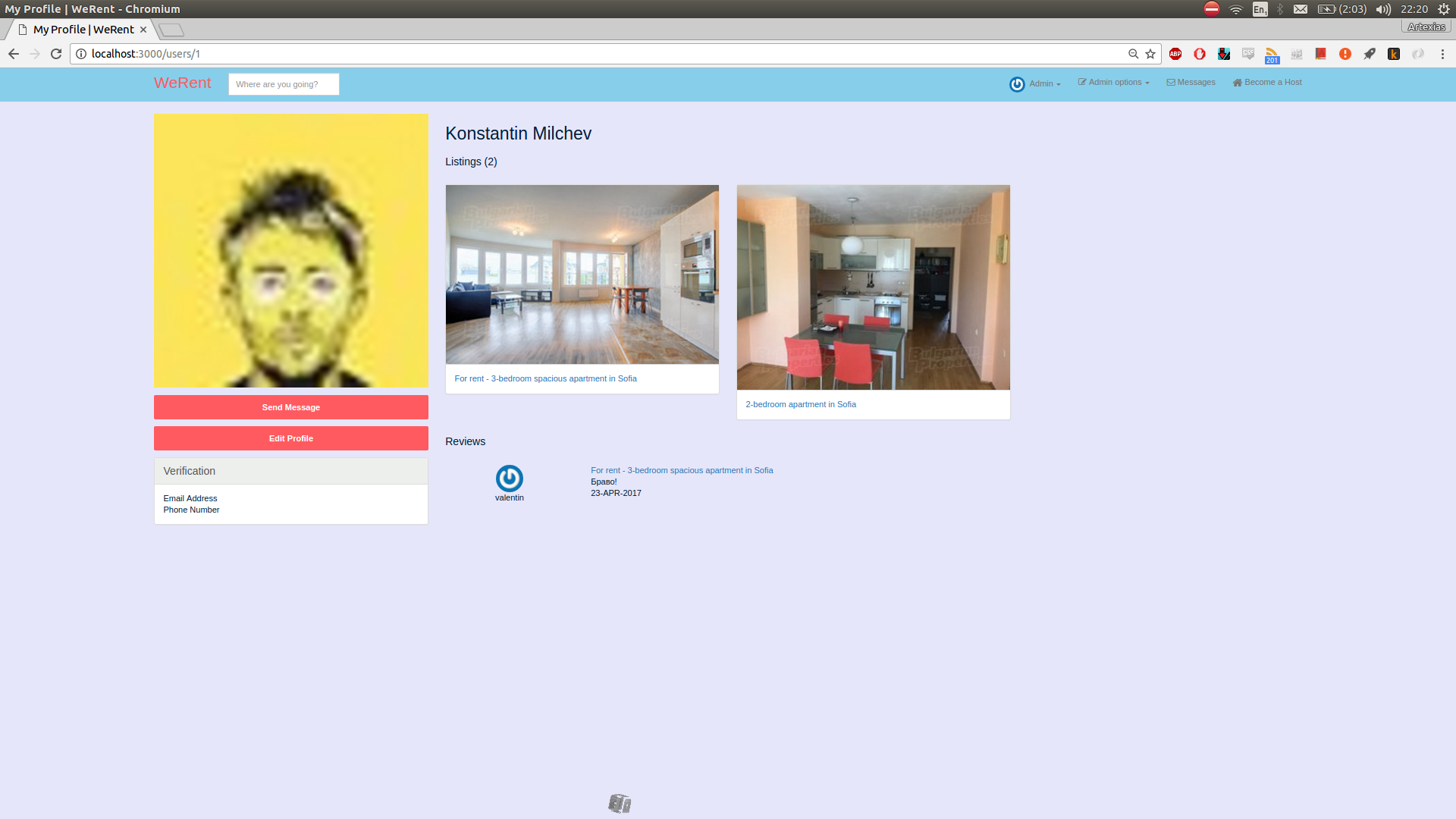
****

**13.** Edit Rooms – 2

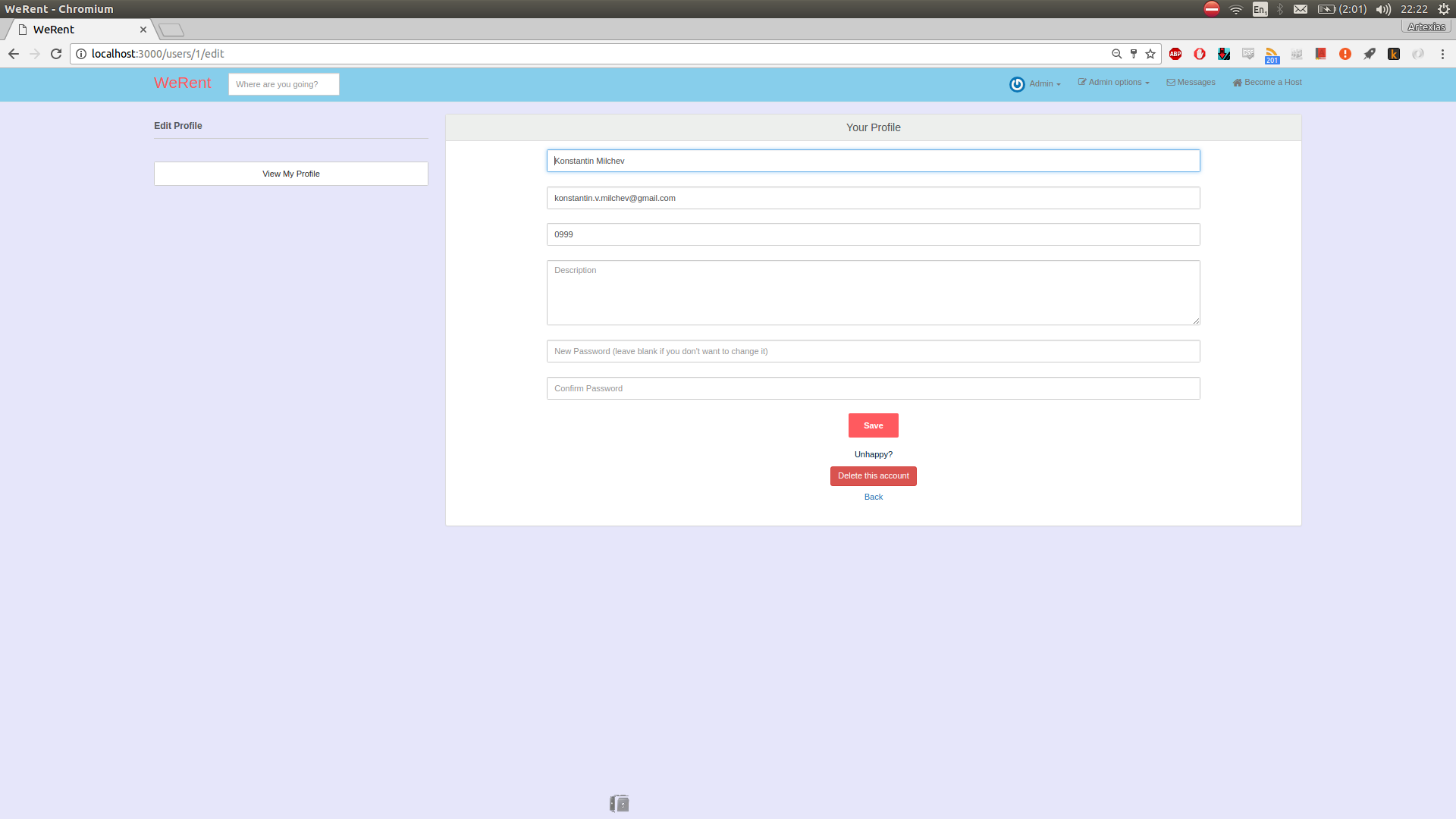
**14.** Edit Users

****

**15.** Edit Users – 2

****

**16.** Edit Users – 3

****

**17.**

1. https://github.com/tylergannon/toastr-rails [↑](#footnote-ref-2)
2. . https://github.com/plataformatec/devise [↑](#footnote-ref-3)
3. . https://github.com/ryanb/private\_pub [↑](#footnote-ref-4)
4. . https://jqueryui.com/datepicker/ [↑](#footnote-ref-5)
5. . https://pixabay.com/en/videos/ [↑](#footnote-ref-6)