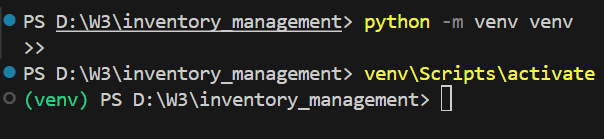
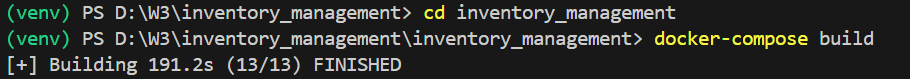
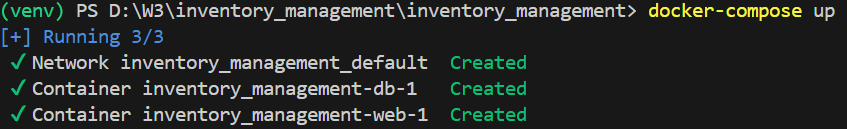
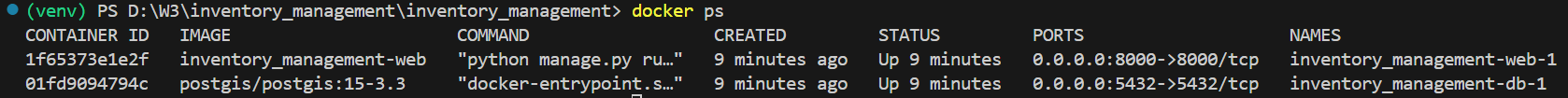
1. **Ensure that the Virtual Environment is Activated**  
   Before proceeding, verify that you have activated the virtual environment. This is crucial to ensure that the correct dependencies are used for the project.

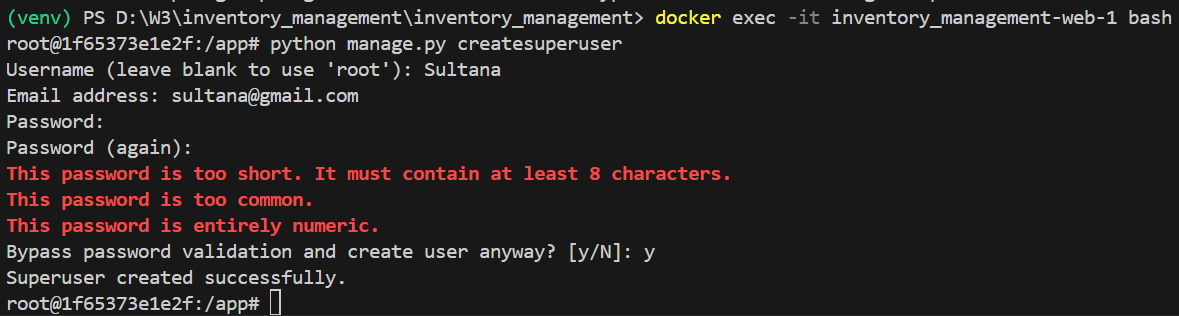






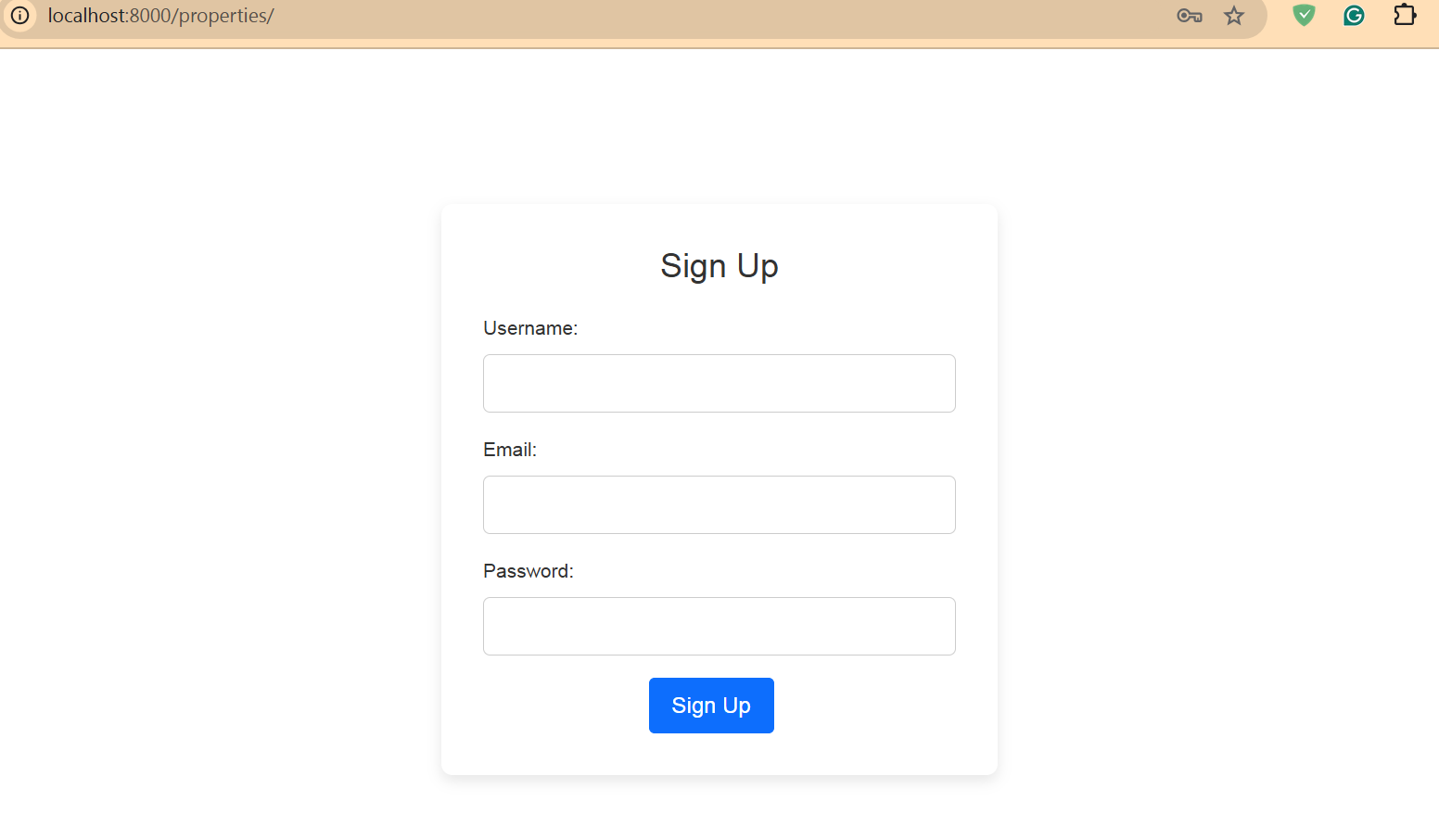


1. **You have to create an admin or superuser in the web container who can manage** **all the tables.**

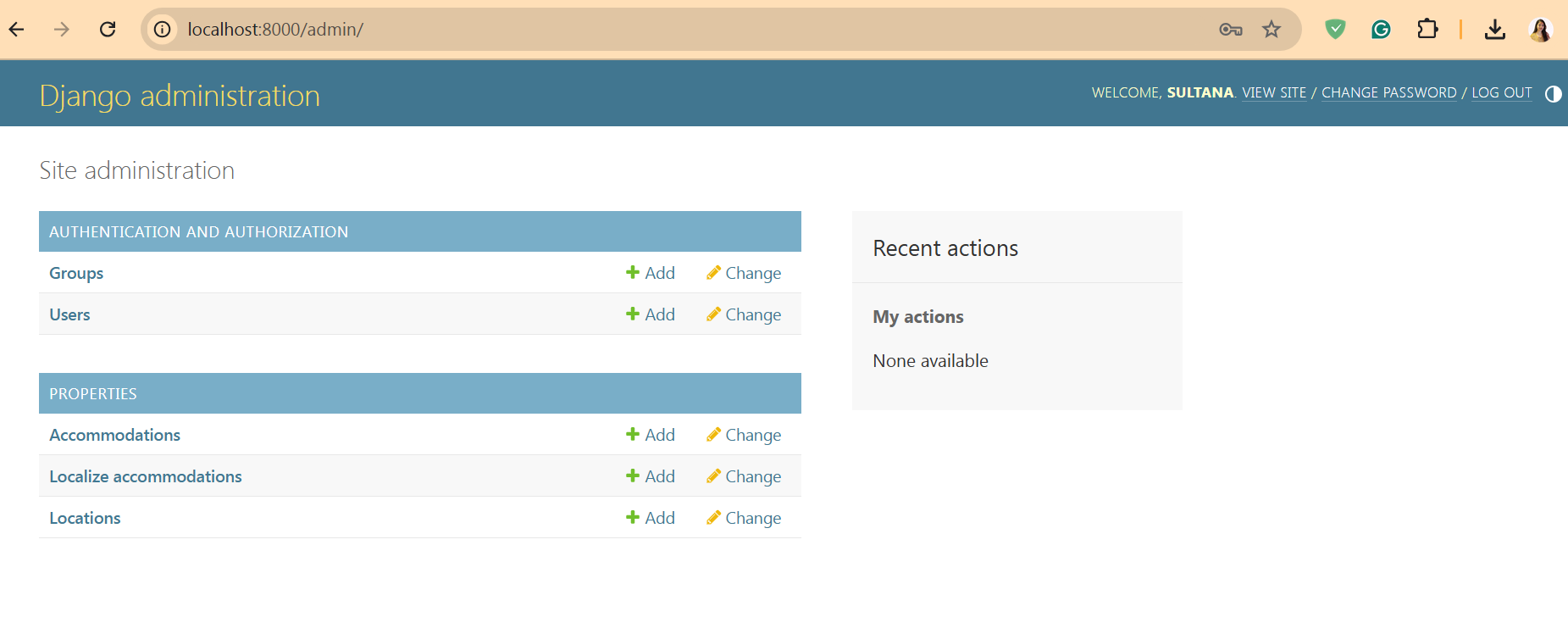


* **Property Owner Sign-Up Page**  
  The **Sign-Up** page at <http://localhost:8000/properties> allows **Property Owners** to register by providing a unique username and email address.
  + Upon successful registration, a confirmation message will be displayed, indicating that the user has been successfully registered.
  + If the provided username or email is already in use or does not meet the required validation criteria, an error message will be shown, specifying the issue (e.g., username or email already taken).
  + This ensures that each **Property Owner** has a unique account within the system.

|  |  |
| --- | --- |
|  |  |

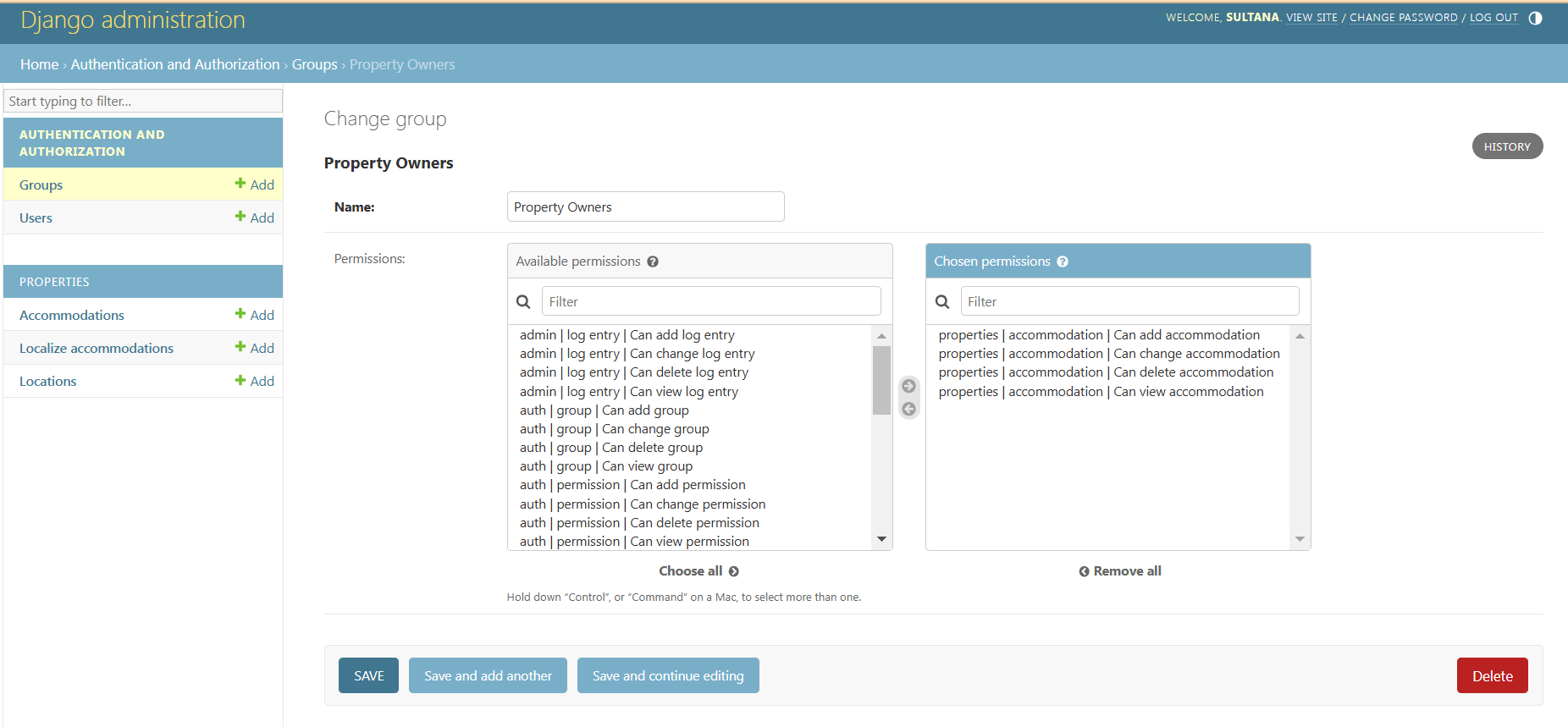


* **Django Admin Interface**  
  The **Django Admin Interface** is accessible at <http://localhost:8000/admin>, where users can log in using their username and password.
  + On your first login, use the **Admin/Superuser** credentials that were created during the setup process.
  + After logging in successfully, you will be presented with an administrative interface that allows you to manage various aspects of the application, such as locations, accommodations, and user permissions.



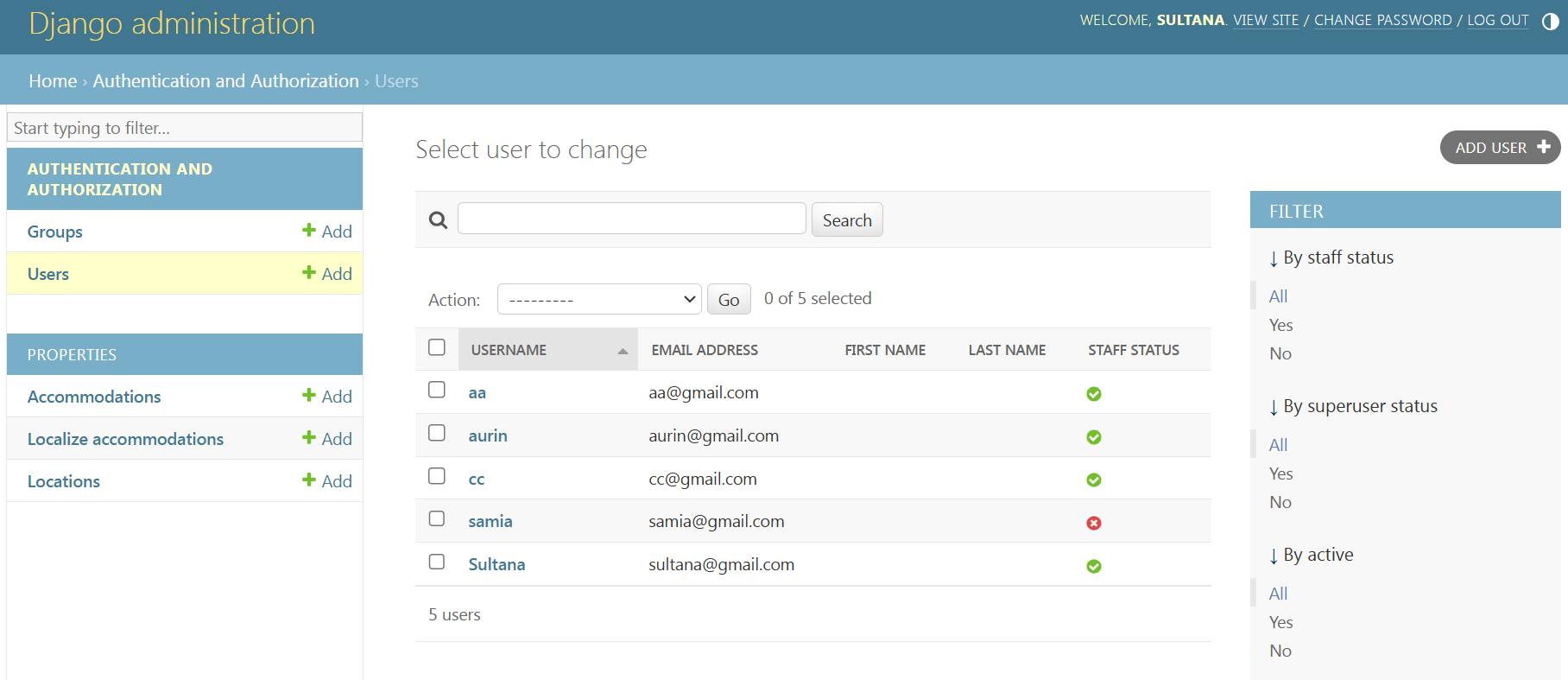
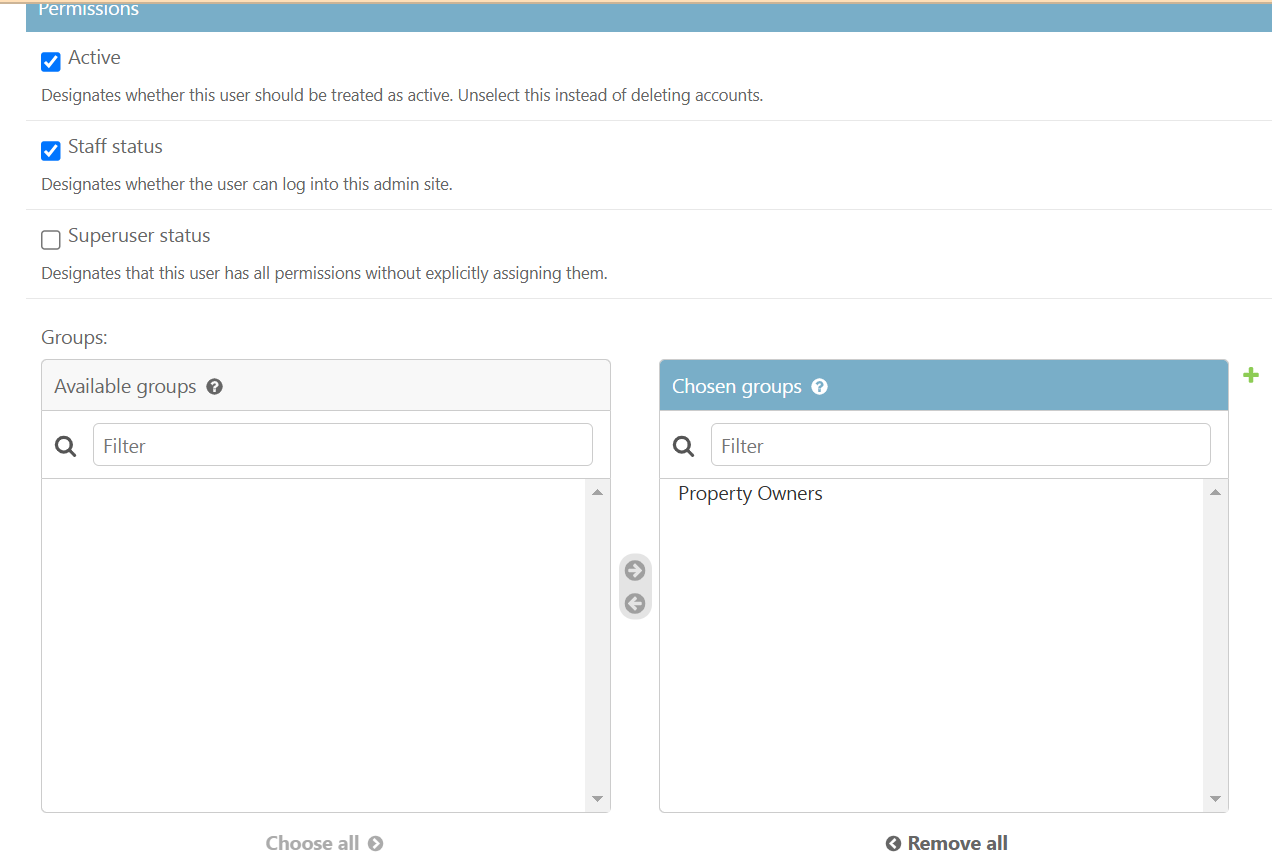
**Managing User Groups and Permissions**

1. **Create Property Owners Group**:
   * In the Django Admin Interface, navigate to **Groups** and create a new group named **Property Owners**.
   * Assign the **Accommodations** table to this group, ensuring that users in this group can only access and manage their own properties.
   * Save the group configuration. As a result, each user in the **Property Owners** group will only be able to view and manage their own accommodations; they will not have access to other users' properties.

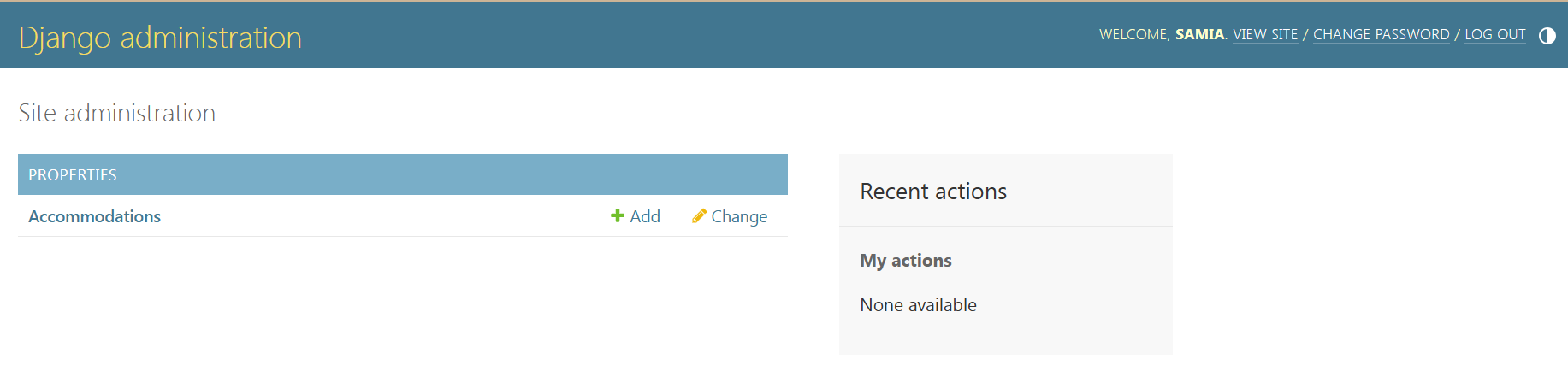


1. **Manage User Access**:
   * Navigate to the **Users** section in the Admin Interface.
   * You will see all registered users, but newly registered users will have a red cross under the **Staff status** column.
   * To enable a user to log in, click on the user, check the **Staff status** box, and save the changes.
   * Only users with the **Staff status** enabled will be able to log into the Admin interface. Users without this status will not be able to access the system.

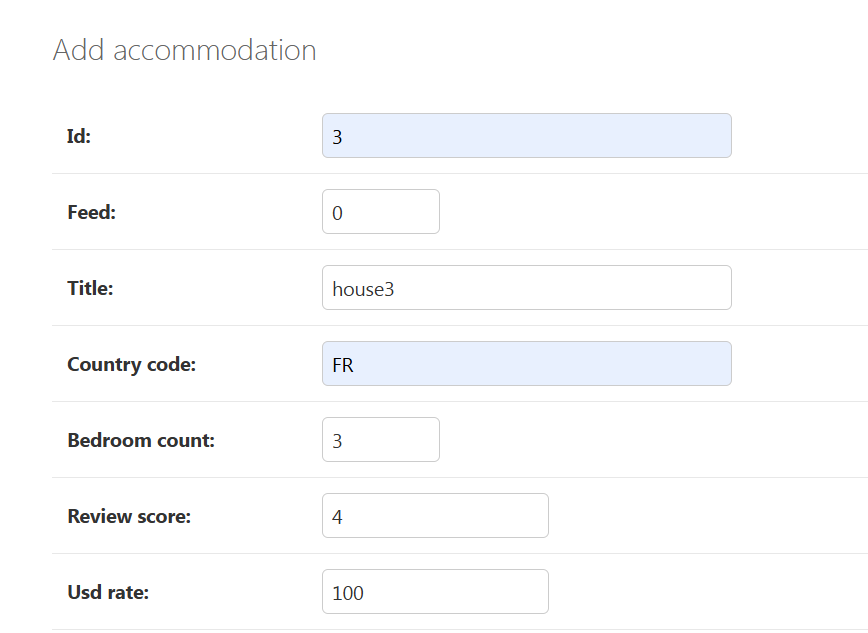
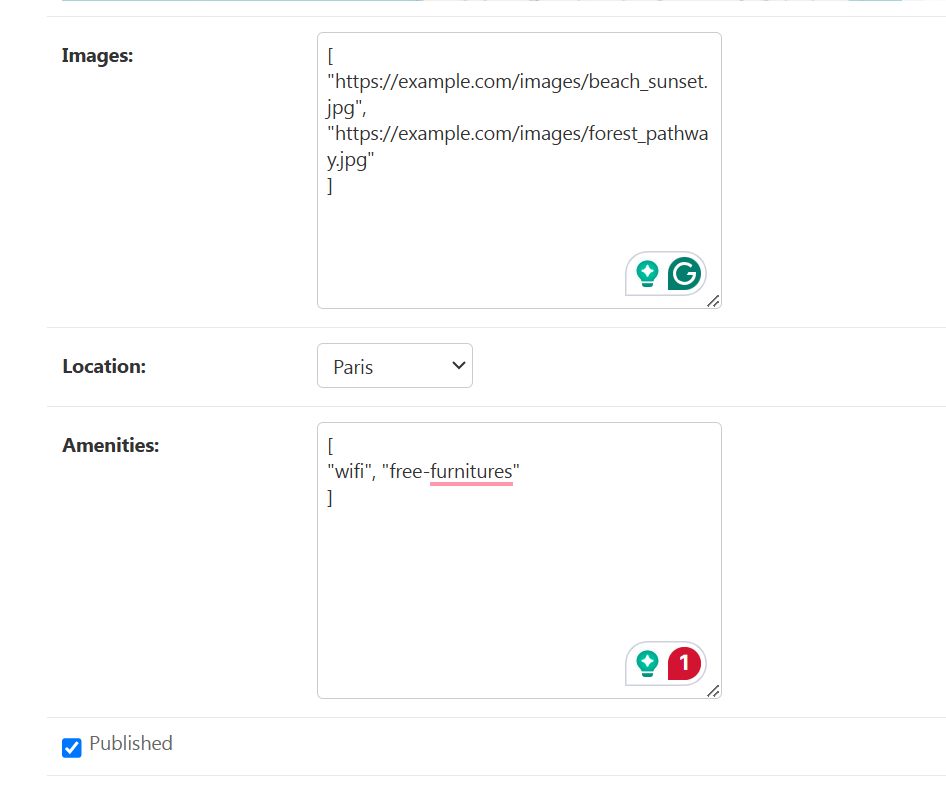
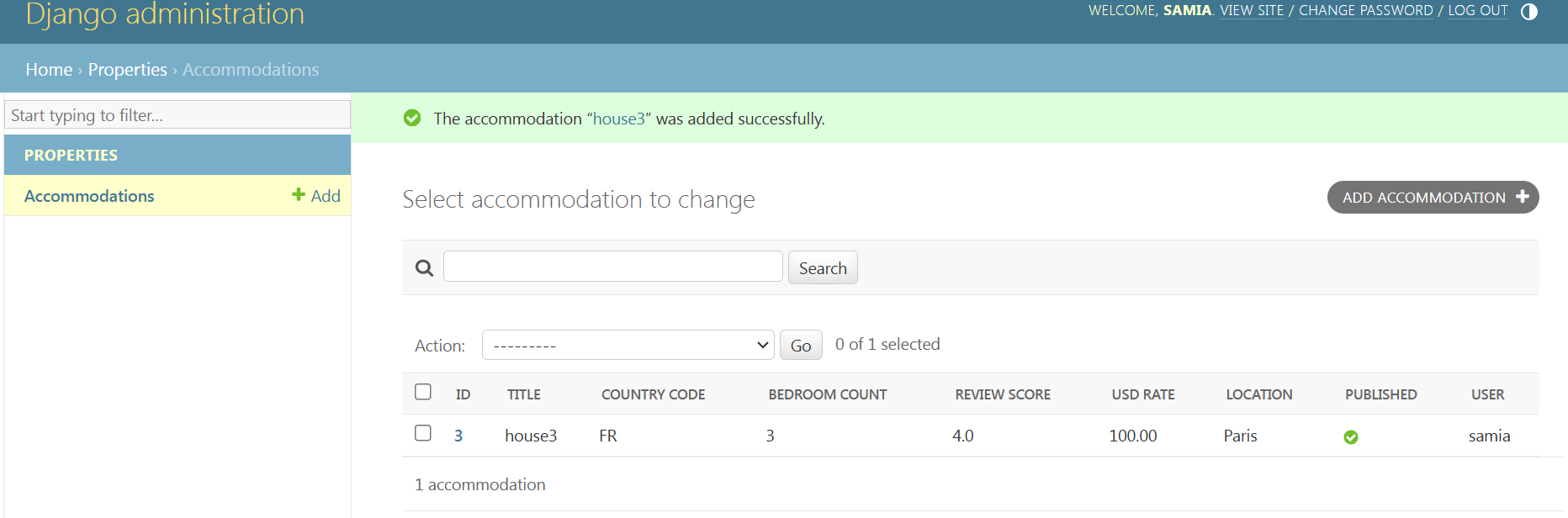
These steps ensure proper user management and access control, providing each user with the appropriate permissions based on their role.

* **Staff Status Activation**:
  + When a new user, such as **Samia**, registers, the **Staff status** will initially be unchecked, represented by a red cross.
  + Once the **Admin** enables the **Staff status** for **Samia**, a green checkmark will appear, indicating that she now has access to the Admin interface.
* **Logging in as the New User**:
  + After the **Admin** has granted staff access, ensure that you **log out** of the Admin interface if you're currently logged in as the Admin.
  + **Samia** can then log in with her credentials at <http://localhost:8000/admin>.
  + Upon logging in, **Samia** will have access only to the **Accommodations** table, where she can manage her own properties. She will not have access to other sections or user data, ensuring role-based access control.



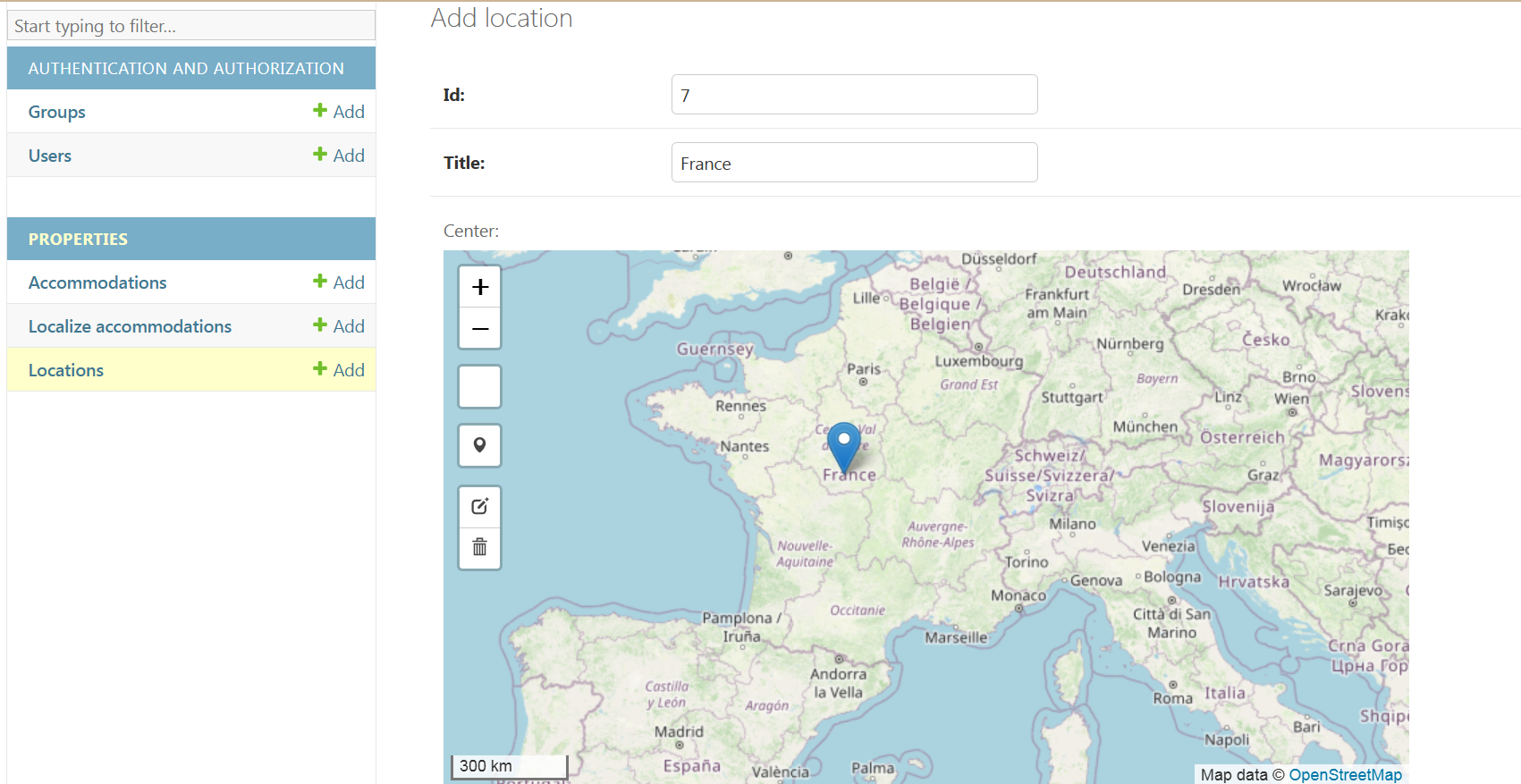
Property Owner adding their property in Accommodations Table

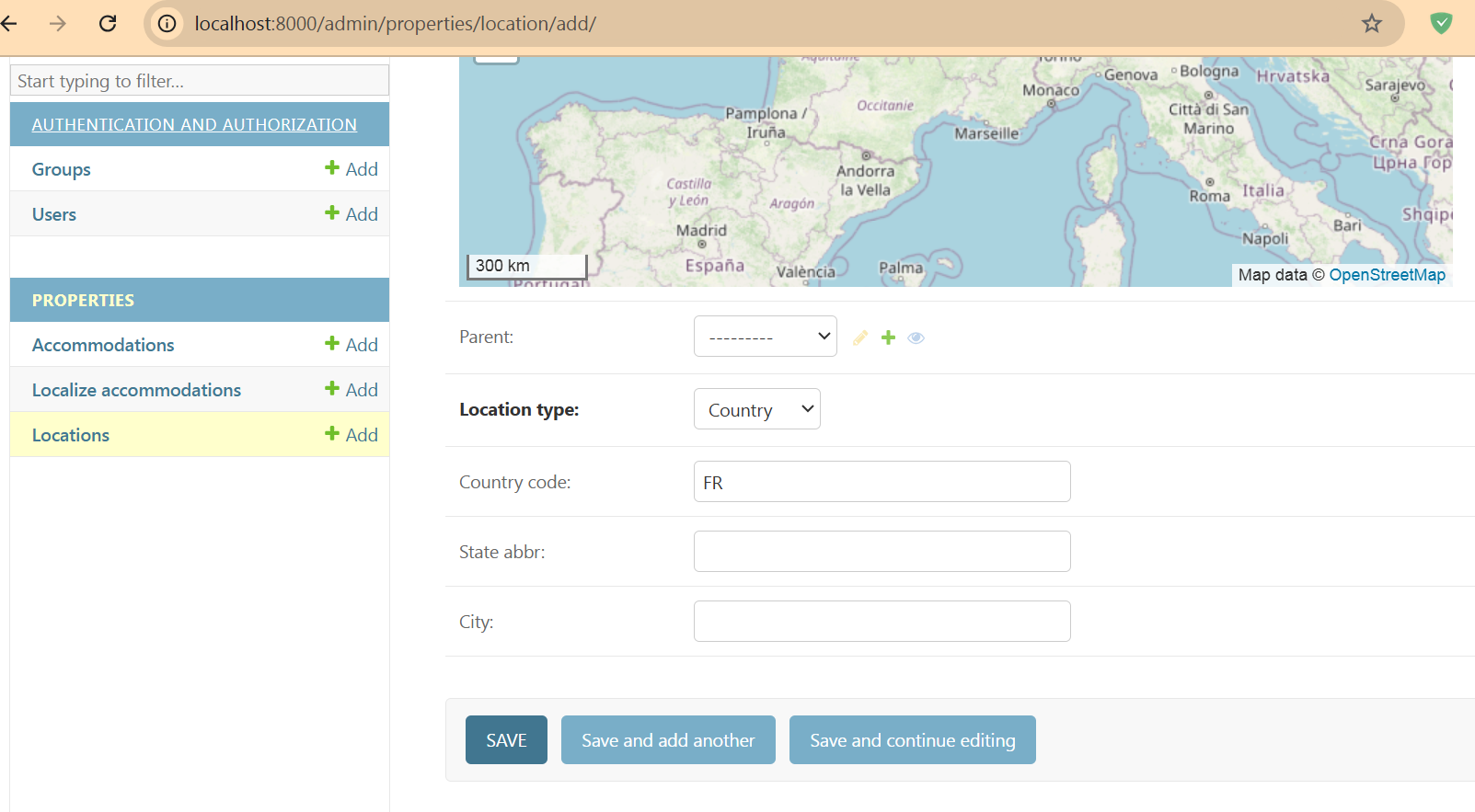
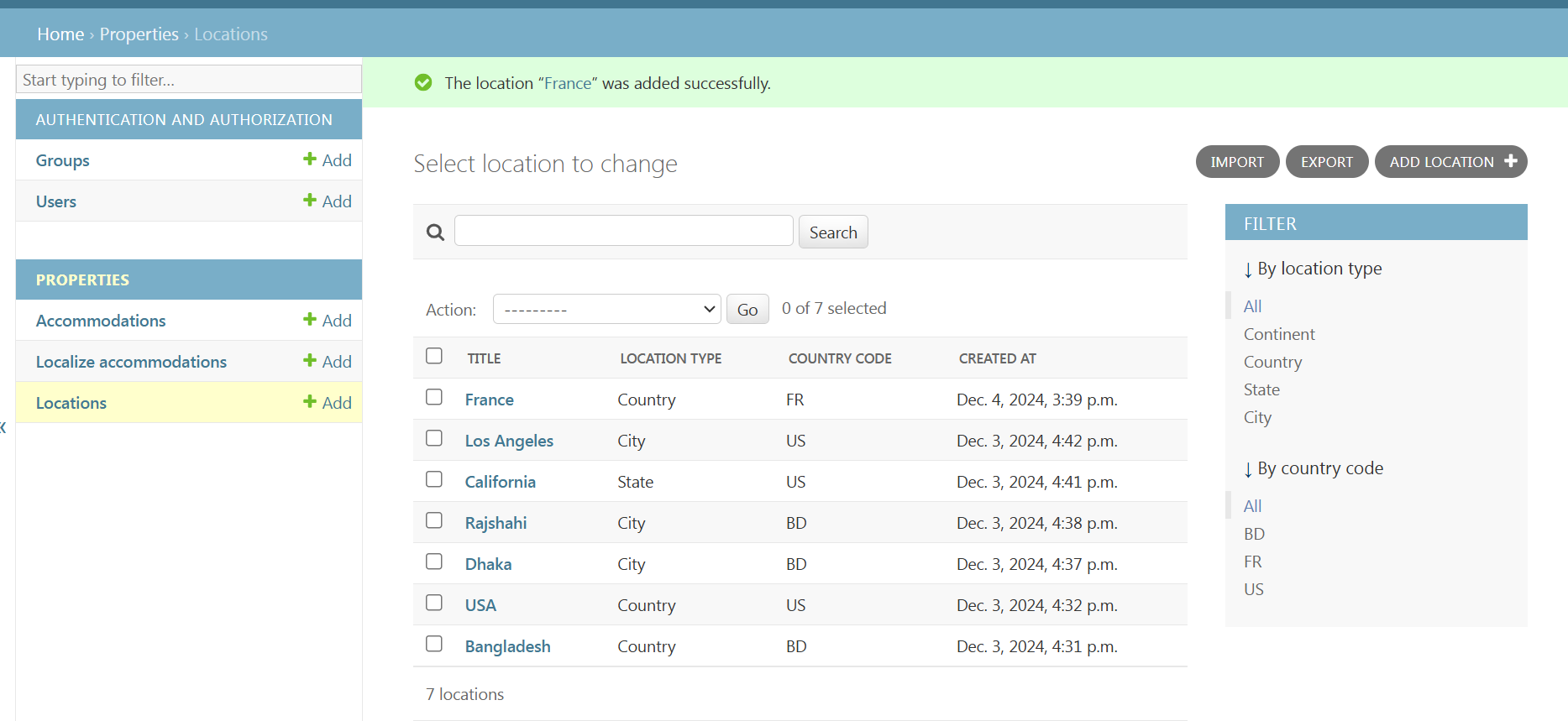
  

**Managing Locations Table**

Only Admin/Superuser has the ability to manage the Locations table. The Location model is designed to handle hierarchical data, including continents, countries, states, and cities.

1. ***Add Countries*:**
   * Start by adding all the countries to the system. For example, to add France as a country:
     + Navigate to the Locations table in the Admin interface.
     + Select Add Location and input France as a Country (Click the green ‘+’ sign)
     + Leave the Parent field empty if France does not require a parent location (or select Continent as its parent if needed).

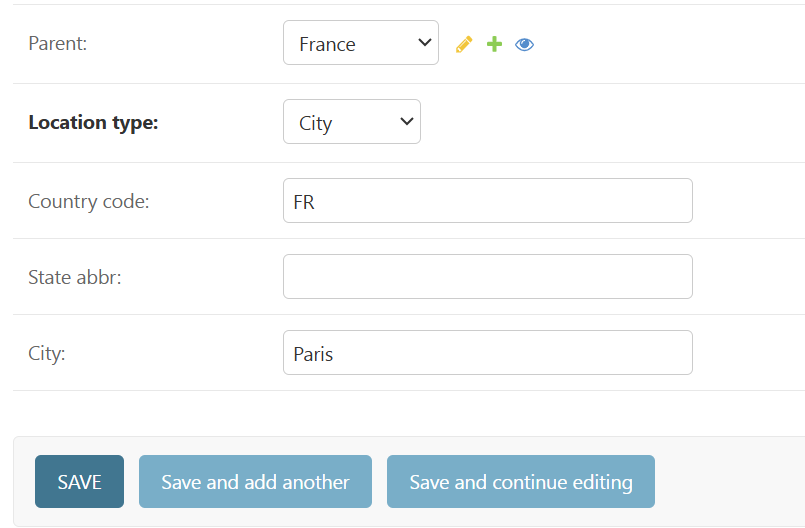
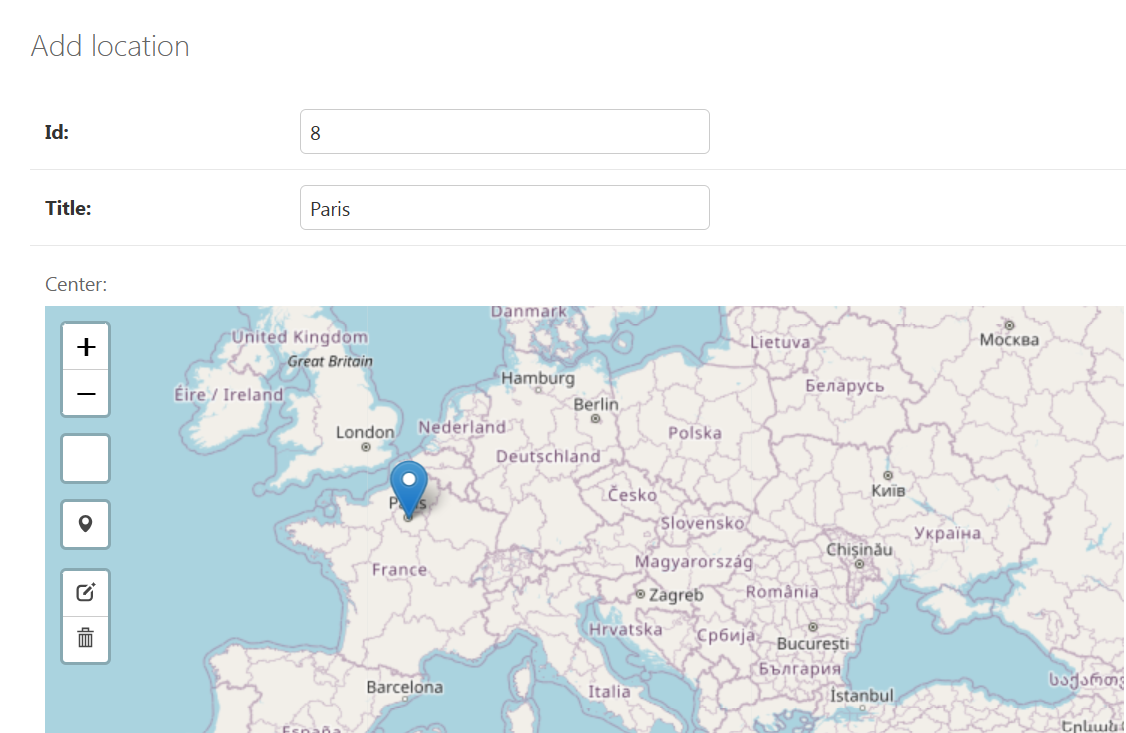


**Countries Without States**:

For countries like **Bangladesh or France**, which do not have states, you can directly add **cities** under the country. In this case, the **Parent** of the city will be the **Country** itself.

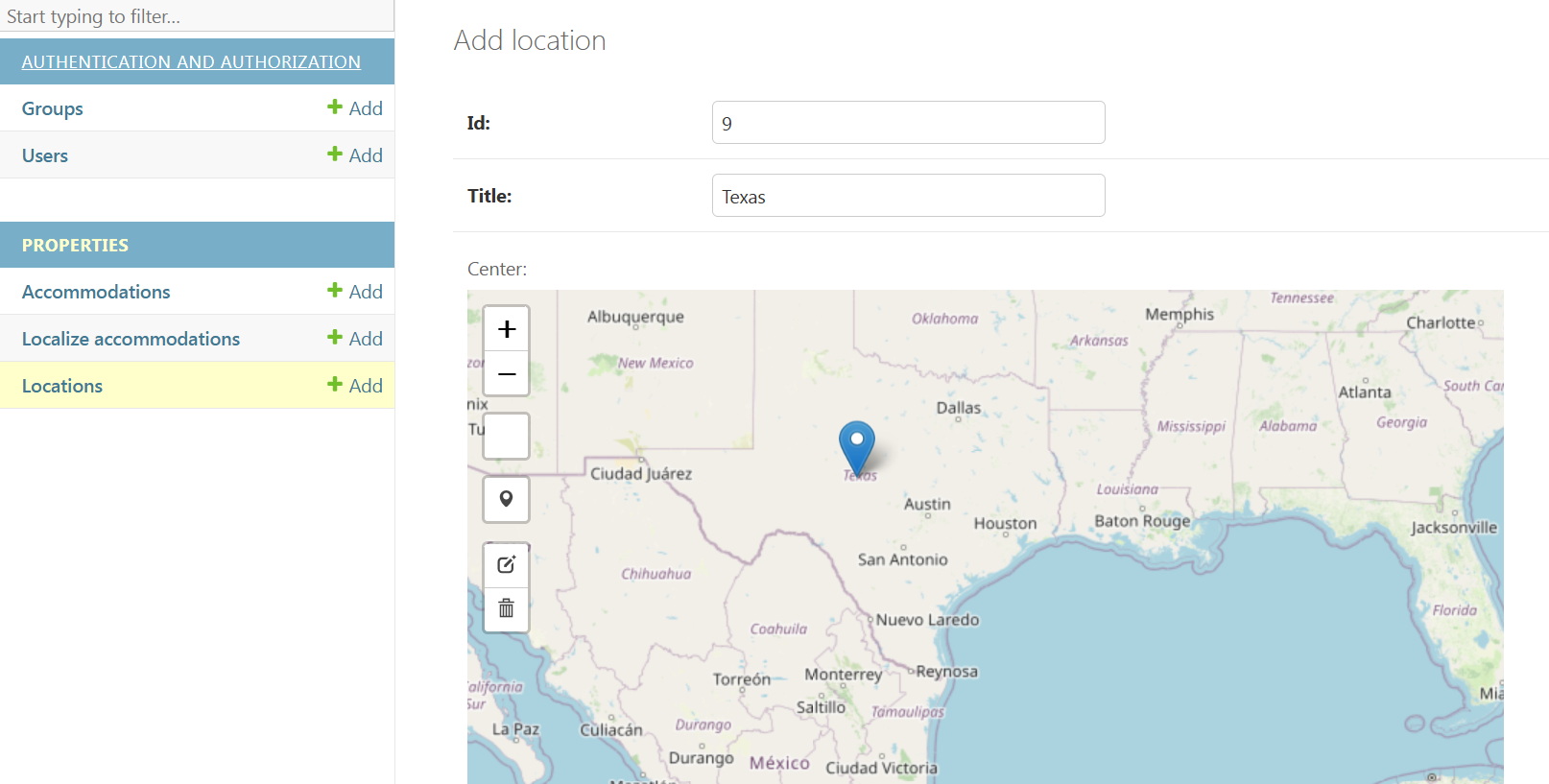
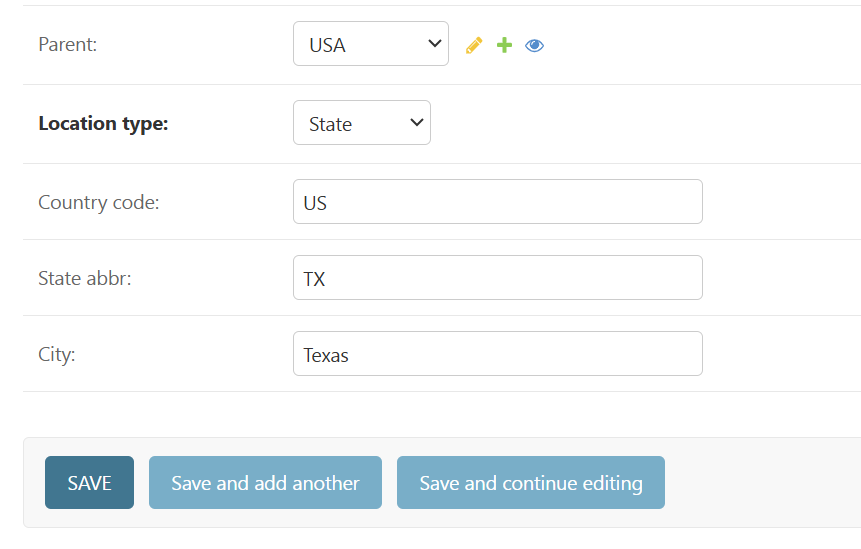
* + For example, add **Paris** as a **City** under **France** with **France** as the **Parent**.
  + The Id will be unique (string, (max 20 characters))

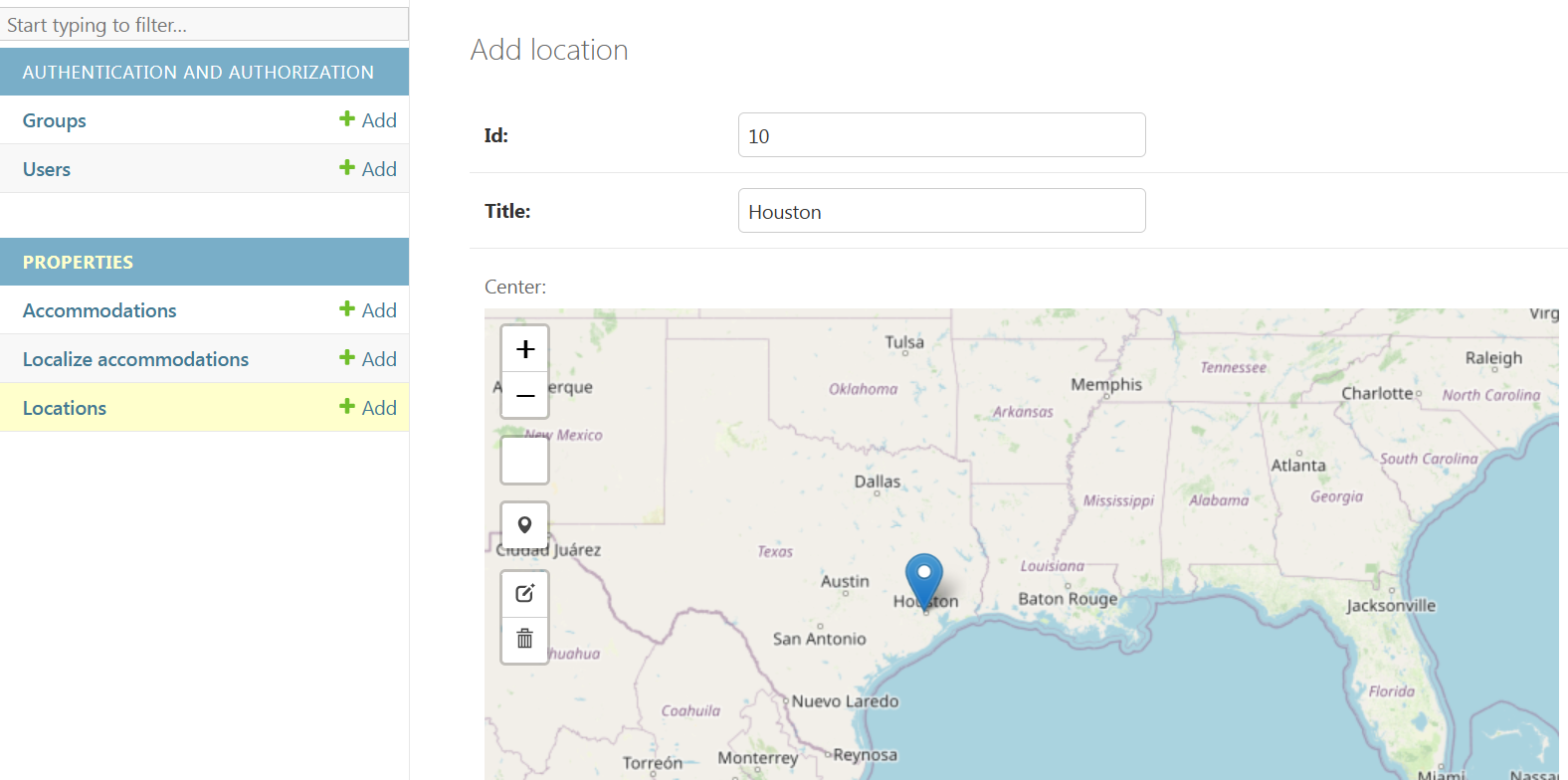
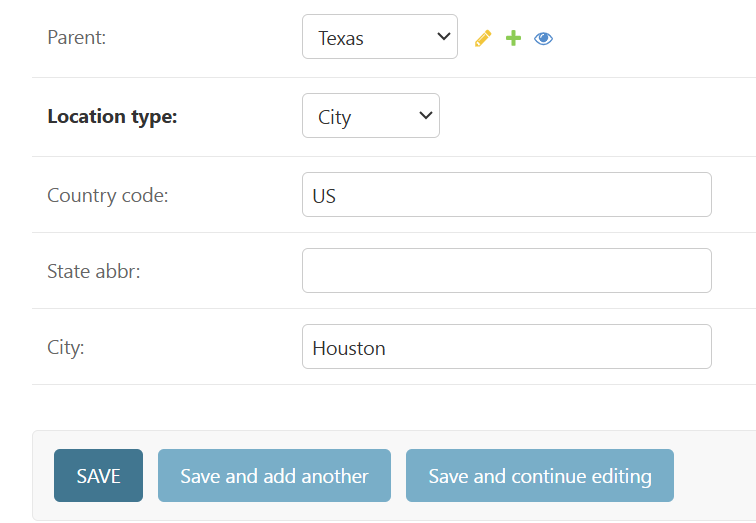


**Countries With States and Cities**:

If the country has states, add each **state** as a separate entry. For instance, for **USA**:

* + Add **Texas** as a **State** under **USA** by selecting **State** as the **Location type** and setting **USA** as the **Parent**.
  + After adding the state, you can then add **cities** under that state. For example, add **Houston** as a **City** with **Texas** as the **Parent**.

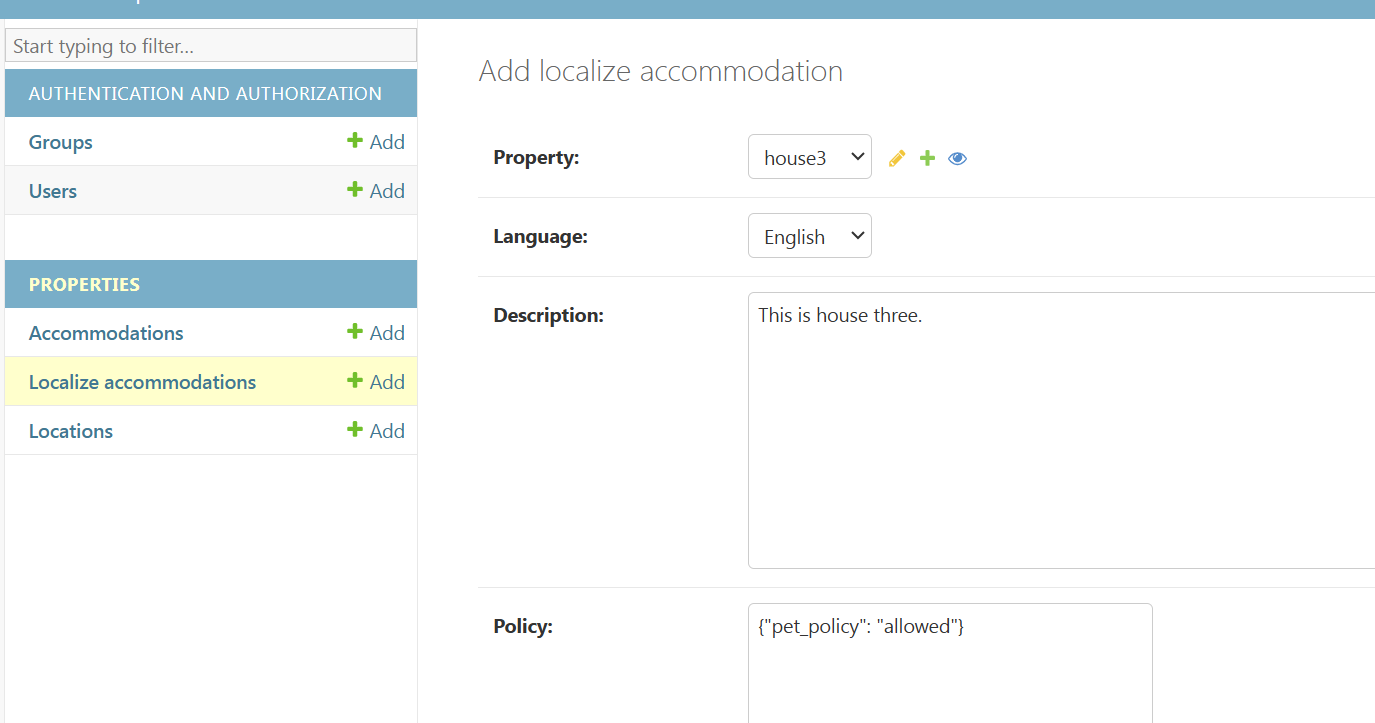
 

1. ***Update & Delete Countries*:**

* Navigate to the **Locations** table in the Django Admin interface.
* Click on the desired entry you wish to update.
* Edit the relevant fields.
* Once you have made the necessary changes, click **Save** to apply the updates.
* In the **Locations** table, select the entry you wish to delete.
* In the **Actions** dropdown menu, choose the **Delete** option.
* Click **Go** to confirm the deletion.
* A confirmation prompt will appear to finalize the deletion. Confirm the action to remove the entry.

**Managing Localize accommodations Table**



**Testing**



