We are using as a central reference point the Listing entity (which has only 3 attributes unique to each listing) because almost all of the data could be linked to it in a very intuitive way.

All the entities have been selected judging on the principle that: if the attributes found in one instance of an entity are repeated more than one time in the data, those attributes must be included in one entity. We have made sure to “package” similar attributes into one entity, as to not increase the complexity of the schema. For the attributes which can be different from one instance to the other, we have set them as relational attributes (to avoid having a new entity all together, and also for not voiding the aforementioned principle).

Note: We included all the attributes of an entity in a big bubble as to conserve space in the diagram. \*\*\*\*MUST SQUIGLY UNDERLINE WEAK ENTITIES\*\*\*\*\*\*\*\*\*

Calendar: there is a ‘one to many’ relation with listing because one listing will have at least one date of availability. The ‘price’ and ‘available’ attributes were set as relational attributes, because they were different for different listings (as I should be)

Accommodation: This entity describes the interior of the listing’s apartment. All of these attributes will change from listing to listing so getting them together in one entity seems to be the right choice. Every listing will have exactly one set of those attributes, hence we used the ‘exactly one’ relation. It can happen that there are more than one listing that have the same set of attributes of this kind, that’s why we opted out on having a weak relation here \*\*See if this can be shown George\*\*

Amenities: The data contained here would make sense to be included in the accommodation entity, but since listings had more than exactly one amenity, we decided to separate it in a separate simple entity.

Policy: Similar to the accommodation entity, we grouped together attributes which are unique to the listing (and also describe the same thing, in this case the rules of the listing) into one place. There is again an ‘exactly one’ relationship for the same reasons. One difference is that there are two relational attributes here (interaction and house\_rules). We decided to put them here because when looking at the data we saw that the policy was similar in most of the listings, with the exception of these two fields. \*\*George show data\*\*

Description: Once again we grouped together attributes which describe the same aspect of the listing. We made this a weak entity, because a description should not exist on it’s own, it should always be paired with a listing.

Review and Reviewer: We decided to go with a ternary relation here between review-reviewer-listing. This is because they will always be used together: A review is given by a reviewer for a listing. The attributes we assigned to each entity are self-explanatory and there is an ‘exactly one” relation for review because that has to be unique to the listing-reviewer. There are ‘many to many’ relations for both listing and reviewer because a listing can have many reviews from different reviewers, and a reviewer can leave many reviews for different listings.

Host: The host entity has all the attributes which describe a host. There is an “exactly one” relation from listing, because a listing should have exactly one host (we are not allowing multiple host to manage the same listing, same as Airbnb does in real life). The host has an ‘at least’ relation with listing because it makes no sense to be a host when you have no listing to manage.

\*\*Maybe remove this\*\*Note: We thought about having a User entity which split into Reviewer and Host by a “Is a” relation, because from a logical point of view it could make sense. But since reviewer and host did not share any similar attributes (apart for name and id), we decided to not do this, as it would unnecessarily increase the complexity of the schema.

Verification: \*\*Why did we split this from host? I don’t remember\*\*

Pricing: For the exact same reasons as with description we decided do have a weak entity of similar data here.

Country, City, Neighborhood: Here we are using the logic that a Country can have multiple Cities, and each City can have multiple Neighborhoods. City and Neighborhood are weak entities to the ones above them because, for example, there can be no city which has no country. Finally, we ‘connect’ the most descriptive of the three (Neighborhood) with Listing with an ‘exactly one’ relation, because each listing is located at exactly one neighborhood. The attributes of country, city and neighborhood are reused for different instances of listings, the ones that change (latitude and longitude), we put inside the relationship, as to be able to keep the property of reusability.