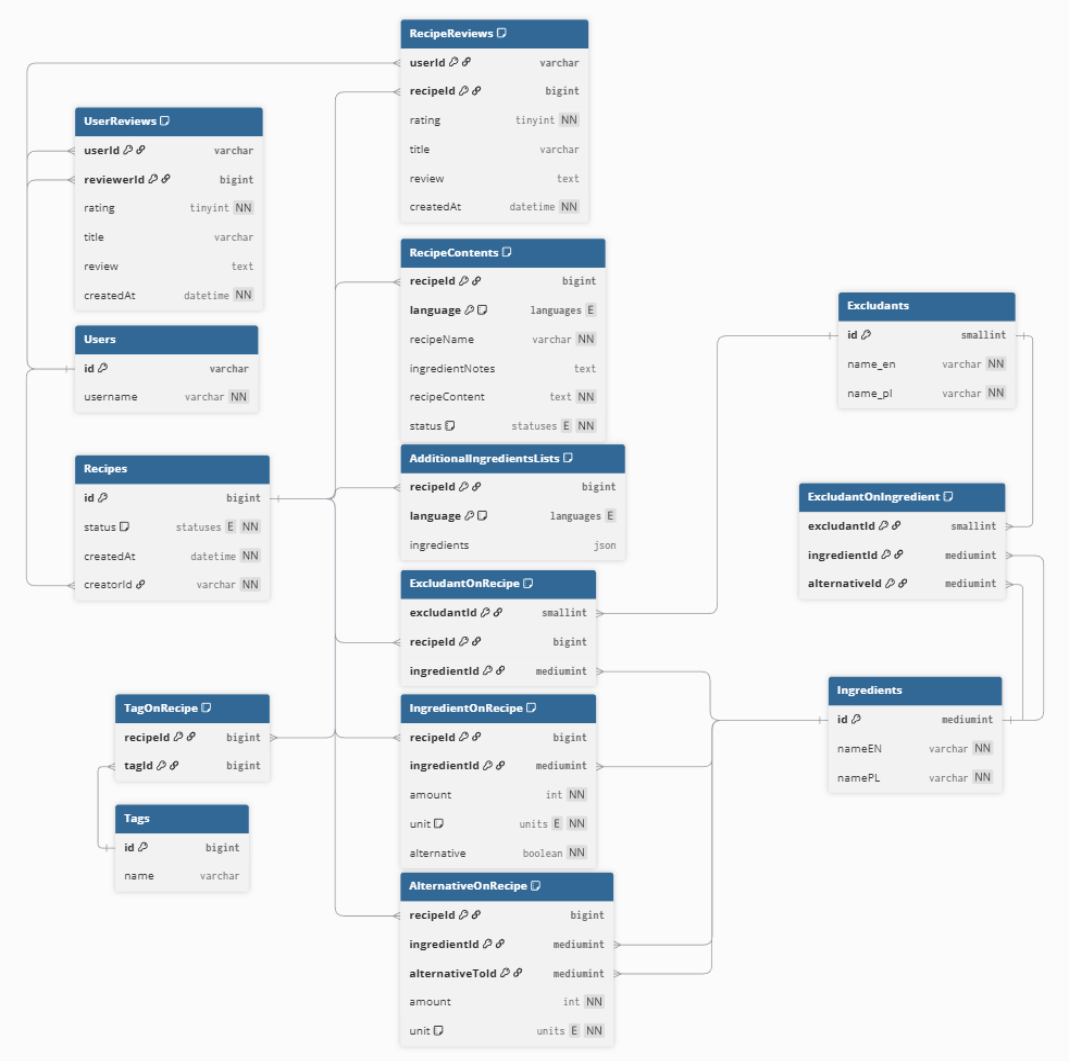
Project “TasteTS”

Database Schema and Implementation

1. Design

The ORM used for the project is Drizzle, and the schema was designed using dbdiagram.io

2. Full schema



The *users* table is simplified, since no changes are planned to the default NextAuth tables. For this reason, other authentication-related tables are omitted.

3. Tables and enums

Here is a short description of the tables, with information about some of the design choices, that were made.

The project includes 3 enums:

- languages – currently with values “polish” and “english”.
- statuses – used for the recipe status, with values “public”, and “private”.
- units – used to represent the amount of each ingredient, with following values:
  - none (e.g. the amount of salt used may be left up to the user by the creator),
  - ml – millilitres,
  - g – grams,
  - pcs – pieces,
  - tsp -teaspoon,
  - tblsp – tablespoon.

5 of the tables can be called “main tables”. These include:

- Users – no changes made from default auth.js.
- Recipes – apart from the id, it has three fields:
  - status – the status of the recipe. The user may create the recipe, and edit it freely, but if the status is not set to “public”, only creator is able to see it. This allows the user to create a recipe, and complete it later, so that other users will only see the final version,
  - createdAt – datetime field set at the time and date of creation of the recipe,
  - creatorId – the id of the user, who created the recipe.

  Two other tables are also tightly associated with *Recipes*. These include:

  - RecipeContents – contains the fields:
    - recipeId – the id of the recipe,
    - language – the language of this version of the recipe, The fields recipeId, and language create a composite primary key,
    - recipeName – the name of the recipe in selected language,
    - ingredientNotes – in this field the creator may leave additional info solely about the ingredients. This may include potential alternatives, or other suggestions,

        - recipeContent – the content of the recipe in selected language,
        - status – the status of the version of the recipe. The user may start creating a different language version of the recipe, and edit it freely, but if the status is not set to “public”, only creator is able to see it. It enables the user to create a version with different language, and finish it later, so that the other users will see only the final version.
    - AdditionalIngredientsLists – temporarily stores ingredients, that are not yet added to the database. Has fields *recipeId*, and *language* following the logic of the table above, and *ingredients* field, which stores data in JSON format.
- Ingredients – apart from the id, it has two fields: *nameEN*, and *namePL*, which have the name of the ingredient in English, and Polish.
- Tags – Tags are added to recipes and are created by users. The table has id, and name fields.
- Excludants – this table stores “flags” which are added to ingredients, and recipes, which may prompt the user to exclude a group of ingredients (e.g. containing gluten, dairy, or meat), from searched recipes. Every excludant is connected to a group of ingredients, and the system allows inclusion of excludant-free alternatives (e.g. lactose-free milk, may replace regular milk in the recipe). The user may disable these alternatives if necessary (e.g. lactose-free milk may have very slightly different taste, than regular milk, which some users may find unacceptable). If the alternative is not present, or it is disabled, the excludant is connected directly to the recipe, to speed up the search process. Similarly to *Ingredients* table, apart from the id, it has two fields: *nameEN*, and *namePL*, which have the name of the excludant in English, and Polish.

Additionally, there is also a *RecipeReviews* table, which contains user reviews of the recipes. userId, and recipeId, which are the ids of the user, and recipe, form a composite primary key. The other fields are: *rating*, *title*, and *review*, which contain the rating of the recipe by the user, the title of the review, and full content of the review.
The table *UserReviews* follows the logic of *RecipeReviews*, however instead of *recipeId* it has a *reviewerId* field.

4. Notes on many-to-many relationship tables

- ExcludantOnRecipe – contains *ingredientId* field, which points at the ingredient, which caused the linkage of excludant to recipe. If multiple ingredients have the same excludant, there are several copies of the same link. The user may add an alternative to only one of the ingredients, and it would cause the deletion of only one excludant link.
- ExcludantOnIngredient – contains *alternativeId* to each ingredient. If there are many alternatives, there exactly as many copies, plus additional one with *alternativeId* set to 0. Some ingredients may not have any alternatives.
- IngredientOnRecipe – has *alternative* field, with type of boolean. Implemented to speed up operations – if set to false, the AlternativeOnRecipe search will be omitted.
- AlternativeOnRecipe – has *alternativeToId* field, which points at an ingredient, which the alternative substitutes. In the rare case of an alternative replacing two ingredients, there are as many copies as needed.

5. Implementation

Full database was implemented with commit **ff41df0**.

Notes:

- Drizzle may create identifier names longer than MySQL's 64-character limit. Providing custom names was found and chosen as a workaround, and relevant information was included in *schema.ts* file.
- Drizzle is set to reach v1.0 soon. The API and schema will need to be refactored once a migration decision has been made.