Routers

We're almost there! The last step is to configure the `FastAPIUsers` object that will wire the user manager, the authentication classes and let us generate the actual **API routes**.

Configure `FastAPIUsers`

Configure `FastAPIUsers` object with the elements we defined before. More precisely:

- `get_user_manager`: Dependency callable getter to inject the user manager class instance. See UserManager.
- `auth_backends`: List of authentication backends. See Authentication.

```
import uuid

from fastapi_users import FastAPIUsers

from .db import User

fastapi_users = FastAPIUsers[User, uuid.UUID](
    get_user_manager,
    [auth_backend],
)
```

Typing: User and ID generic types are expected

You can see that we define two generic types when instantiating:

- `User`, which is the user model we defined in the database part
- The ID, which should correspond to the type of ID you use on your model. Here, we chose UUID, but it can be anything, like an integer or a MongoDB ObjectID.

It'll help you to have **good type-checking and auto-completion**.

Available routers

This helper class will let you generate useful routers to setup the authentication system. Each of them is **optional**, so you can pick only the one that you are interested in! Here are the routers provided:

- Auth router: Provides `/login` and `/logout` routes for a given authentication backend.
- Register router: Provides `/register` routes to allow a user to create a new account.
- Reset password router: Provides `/forgot-password` and `/reset-password` routes to allow a user to reset its password.
- Verify router: Provides `/request-verify-token` and `/verify` routes to manage user e-mail verification.
- Users router: Provides routes to manage users.
- OAuth router: Provides routes to perform an OAuth authentication against a service provider (like Google or Facebook).

You should check out each of them to understand how to use them.