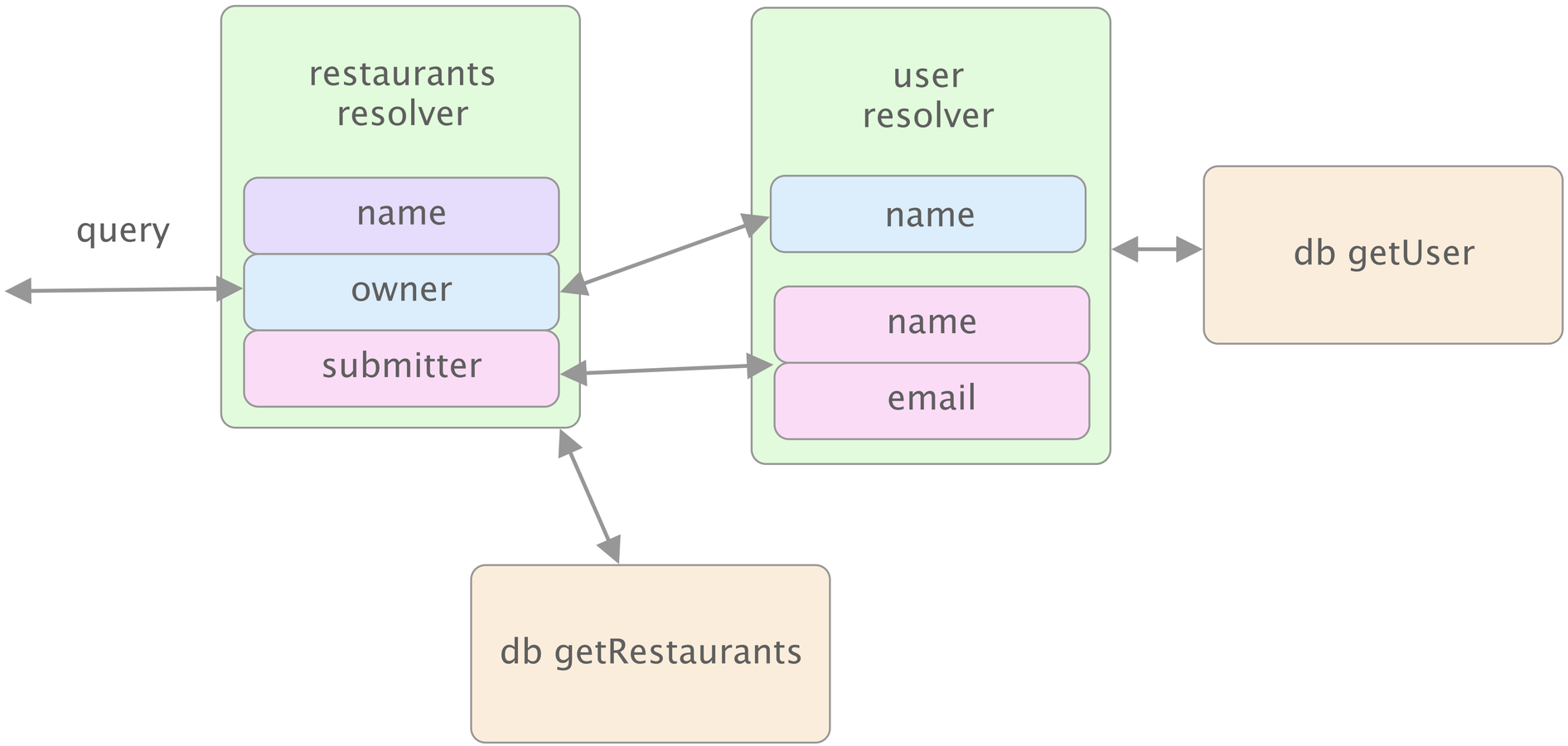
Authorization

How to handle Authorization?

Depends on the app!
Could be all the same authorisation, but mostly has to be decided on case by case basis.

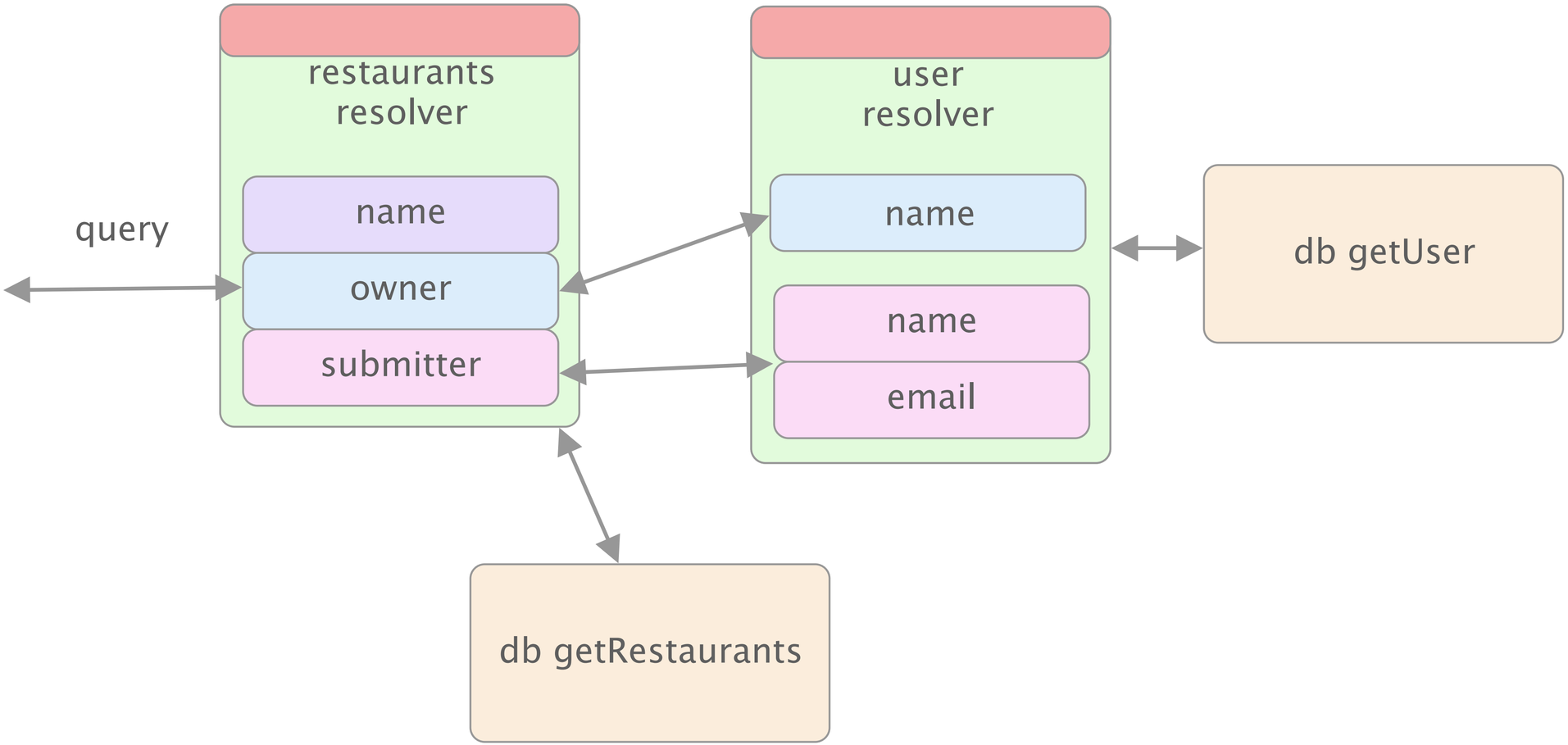
How to handle Authorization?

```
{
  restaurants {
    name
    owner {
      name
    }
    submitter {
      name
      email
    }
  }
}
```



Resolvers

```
{
  restaurants {
    name
    owner {
      name
    }
    submitter {
      name
      email
    }
  }
}
```

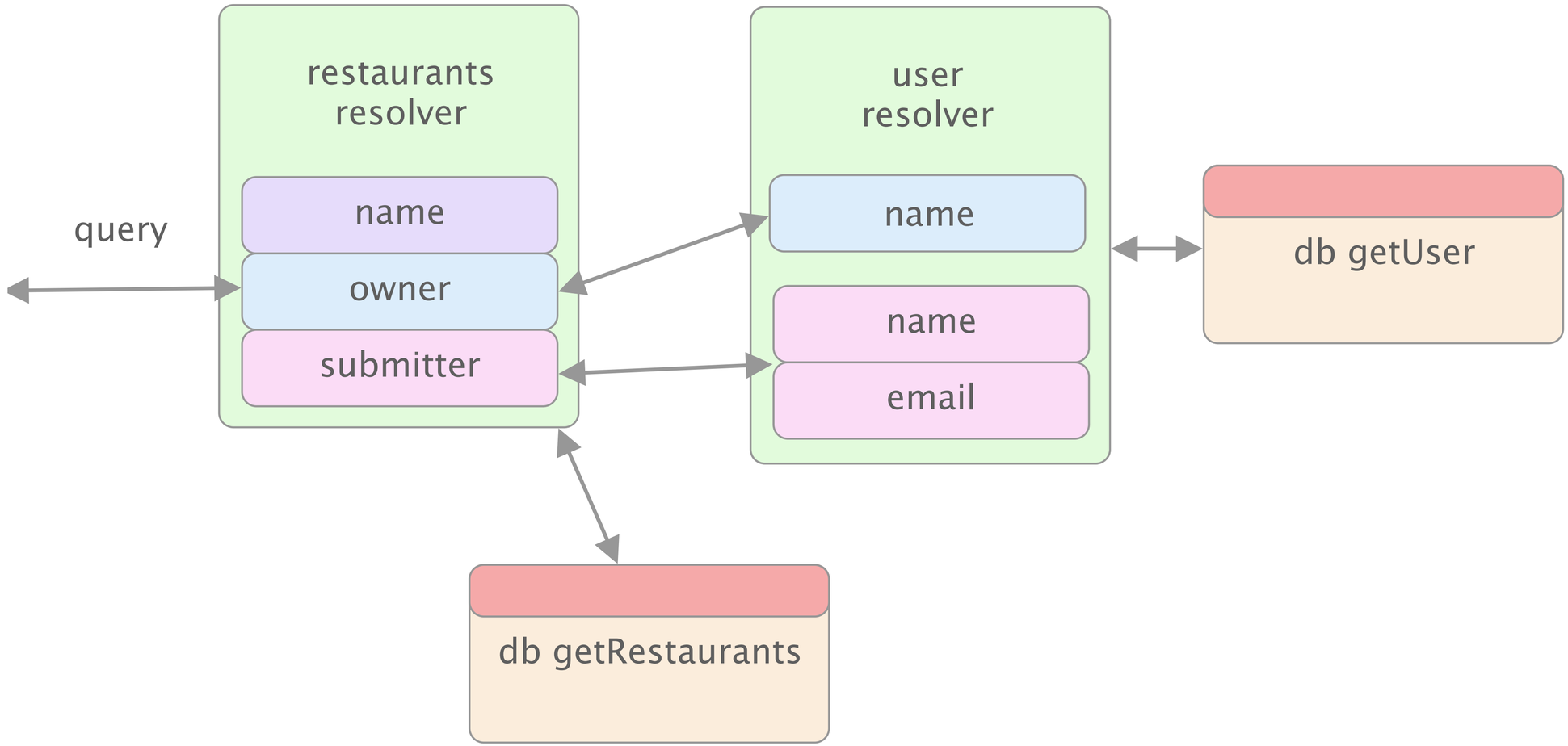


```
const { ForbiddenError } = require("apollo-server")

module.exports = {
  Mutation: {
    addFruitToBasket: (_parent, args, context) => {
      if (!context.user)
        throw new ForbiddenError("Please login first")
      return addFruitToBasket({ …args, user });
    },
  }
};
```

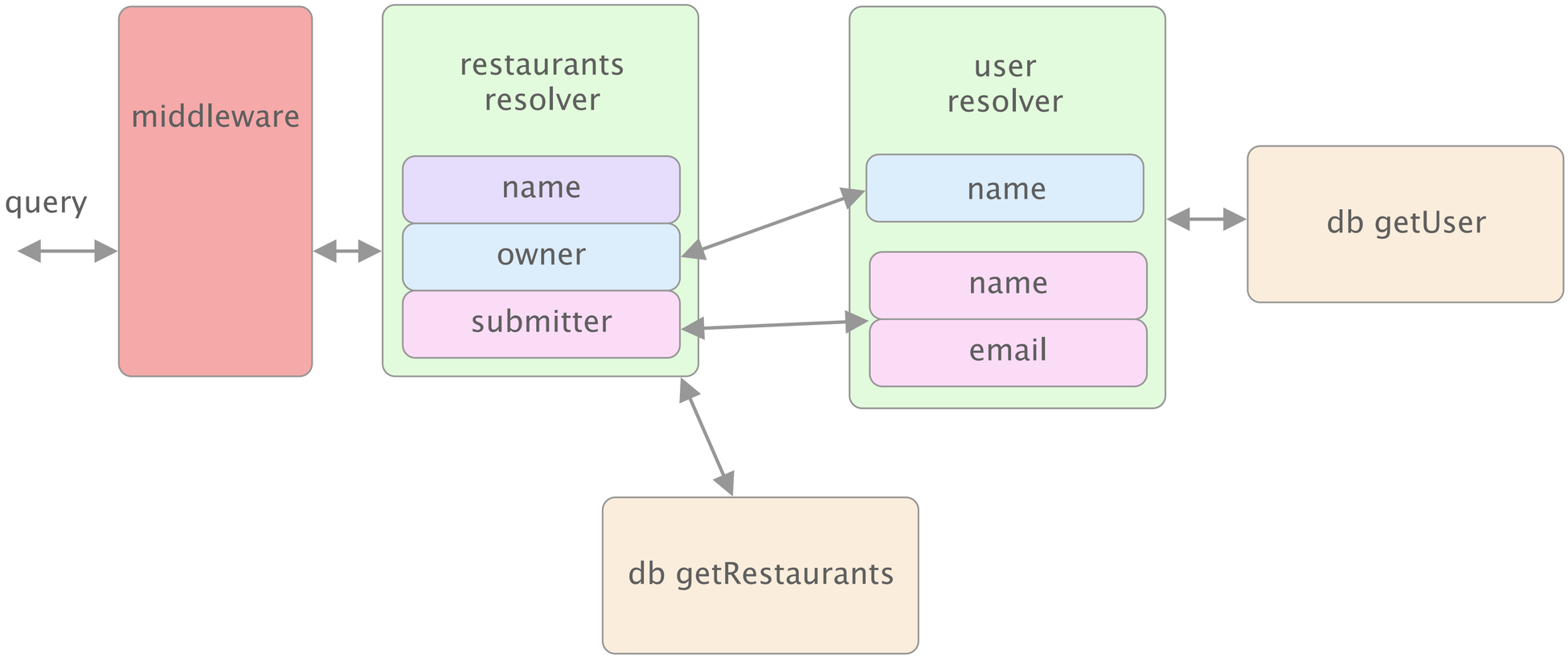
Models/REST Layer

```
{
  restaurants {
    name
    owner {
      name
    }
    submitter {
      name
      email
    }
  }
}
```



Middleware

```
{
  restaurants {
    name
    owner {
      name
    }
    submitter {
      name
      email
    }
  }
}
```



```
const { rule, shield, and, or, not } = require("graphql-shield");
const { applyMiddleware } = require("graphql-middleware");

const isAuthenticated = rule()(async (parent, args, ctx) => ctx.user !== undefined);
const isAdmin = rule()(async (parent, args, ctx) => ctx.user.role === "admin");
const isEditor = rule()(async (parent, args, ctx) => ctx.user.role === "editor");

const permissions = shield({
  Query: {
    frontPage: not(isAuthenticated),
    fruits: and(isAuthenticated, or(isAdmin, isEditor)),
    customers: and(isAuthenticated, isAdmin)
  },
  Mutation: {
    addFruitToBasket: isAuthenticated
  },
  Fruit: isAuthenticated,
  Customer: isAdmin
});

const schema = applyMiddleware(schema, permissions);
```

Exercise

Secure the createRestaurant mutation so only a logged in user can do the mutation.

Do it once within a resolver and then do a second variation using GraphQL Shield

See code hint for on the next Page

```
const { ApolloServer, makeExecutableSchema } = require("apollo-server");
const { importSchema } = require("graphql-import");
const { applyMiddleware } = require("graphql-middleware");
const resolvers = require("./resolvers");
const permissions = require("./permissions");

const typeDefs = importSchema("./schema/index.graphql");
const { getUserByToken } = require("./db/users");

const schema = makeExecutableSchema({
  typeDefs,
  resolvers
});

const server = new ApolloServer({
  schema: applyMiddleware(schema, permissions),
  context: ({ req }) => {
    const token = req.headers.authorization
      ? req.headers.authorization.replace("Bearer ", "")
      : undefined;
    const user = getUserByToken(token);
    return { user };
  },
  tracing: true
});
```

Exercise

During creation of the Restaurant add the logged in user as submitter. Make sure only the current user can see the email.