# 33. Resolver user.ts / Mutation - forgotPassword()

#backend #graphql #resolver #authentication #mutation #email #nodemailer

## Install nodemailer on the server

- nodemailer is a package for sending test emails in development, which can be set up to also use other email providers, e.g. gmail, etc.
- nodemailer.com

**on the server**

```
yarn add nodemailer
yarn add -D @types/nodemailer
```

## Implement sendEmail() utility function

- **Implementeed based on the example on nodemailer.com**

**server/src/utils/sendEmail.ts**

```
import nodemailer from "nodemailer";

export async function sendEmail(to: string, subject: string, text: string) {
  // run with createTestAccount() once, console.log the account, get the password
and use the same account afterwards
  //let testAccount = await nodemailer.createTestAccount();
  //console.log("testAccount: ", testAccount);

  const transporter = nodemailer.createTransport({
    host: "smtp.ethereal.email",
    port: 587,
    secure: false, // Use `true` for port 465, `false` for all other ports
    auth: {
      user: "i3sabqkaflvfyrhh@ethereal.email", //testAccount.user,
      pass: "BS85g71pdwCQDEz5Bz", //testAccount.pass,
    },
  });
```

```
  // send mail with defined transport object
  const info = await transporter.sendMail({
    from: '"Fred Foo 👻" <foo@ethereal.email>',
    to,
    subject,
    html: text,
  });

  console.log("Message sent: %s", info.messageId);
  console.log("PreviewURL: %s", nodemailer.getTestMessageUrl(info));
}
```

---

## Install uuid on the server for token creation

- uuid will be used for creating unique identifiers to be used as a token
- we will store the token in the redis store with expiration time 1 day

```
yarn add uuid
```

/resolvers/user.ts

```
import v4 from "uuid"
```

- Note that it's a good practice to put a prefix in front of the tokens/keys when saving them into redis store so that during development they can be identified easily

constants.tsx

```
export const FORGOT_PASSWORD_PREFIX = "forgot-password:";
```

---

## Add forgotPassword mutation

- Update the user resolver, adding the forgotPassword() mutation, as shown below:
- We basically send the token to user in an email, and when user sends it back to us we look it up in the redis store and if it's valid we let them change password

/resolvers/user.ts

```
@Mutation(() => Boolean)
  async forgotPassword(
    @Arg("email") email: string,
```

```
    @Ctx() { em, redis }: MyContext
  ): Promise<Boolean> {
    const user = await em.findOne(User, { email });
    if (!user) {
      // the email is not in the db
      return true; //  don't let the person know that the email is not in the db
    }

    const token = v4(); // token for resetting pw

    // save token to redis with value userId, expires in 1 day
    await redis.set(
      FORGOT_PASSWORD_PREFIX  + token, // redis key
      user.id,                         // value
      "ex",                            // expiry mode
      1000 * 60 * 60 * 24              // expiration duration - 24 hours
    );
    const resetLink = `<a href="http://localhost:3000/change-
password/${token}">Reset password</a>`;

    sendEmail(email, "Reset Password", resetLink);
    return true;
  }
```