\*\*Skapa en router.get() för att kunna läsa list av data till ex. comments:

skicka in alla information som server behöver för att kunna skapa dynamisk pagination.

till ex. item\_per\_page, page, nextpage osv. (i lektionen kommer jag förklara varje del)

router.get("/comment", async (req, res) => {

const Item\_per\_page = 2;

//sida nummer från queries

const page = +req.query.page|| 1;

//totalComment från db kommer att tilldelas här senare

let totalComment;

//hämtar total comments

const commentsCount = await Comment.find()

.countDocuments()

console.log(commentsCount);

//totalt comment har tilldelats

totalComment = commentsCount;

//hämtar alla comments info

const comments = await Comment.find()

.skip((page-1)\* Item\_per\_page)

.limit(Item\_per\_page)

//.sort({author:req.query.sort})

// { comment : []}

//comments:comments

//console.log(req.query.sort)

res.render("comment",

{comments,

totalComment,

//när itemperpage \*page är mindre än totalt comment har vi nästa sida

hasNextPage: totalComment > Item\_per\_page\*page,

//Item\_per\_page\*page <totalComment,

hasPreviousPage: page >1,

nextPage: page+1,

previousPage: page-1,

currentPage:page,

lastPage:Math.ceil(totalComment/ Item\_per\_page)

});

}

)

i comment.ejs :

<% if(comments.length >=1) { %>

<div class="pagination">

<% if(currentPage !==1 && previousPage !==1) { %>

<a href="/comment/?<%=page=1 %>">1</a>

<% } %>

<% if(hasPreviousPage) { %>

<a href="/comment/?page=<%=previousPage %>" > <%= previousPage %> </a>

<% } %>

<a href="/comment/?page=<%=currentPage %>"> <%= currentPage %> </a>

<% if(hasNextPage) { %>

<a href="/comment/?page=<%=nextPage %>" > <%= nextPage %> </a>

<% } %>

<% if(lastPage !==currentPage && nextPage !==lastPage) { %>

<a href="/comment/?page=<%=lastPage%>" > <%= lastPage %> </a>

<% } %>

<% }%>

</div>

</div>

Alternativ:

<div>

<%var counter = 1 %>

<% for(var i= product\_per\_page; i< countProduct; i+=product\_per\_page) { %>

<a href="/products/?page=<%=counter%>"><%=counter%></a>

<%

counter++;

%>

<% } %>

</div>

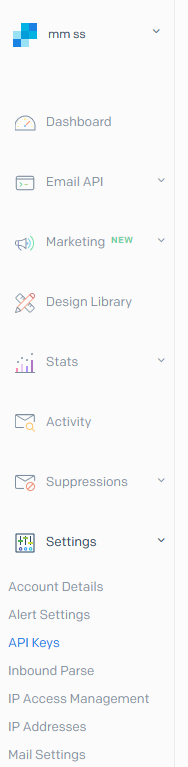
Sending mail:

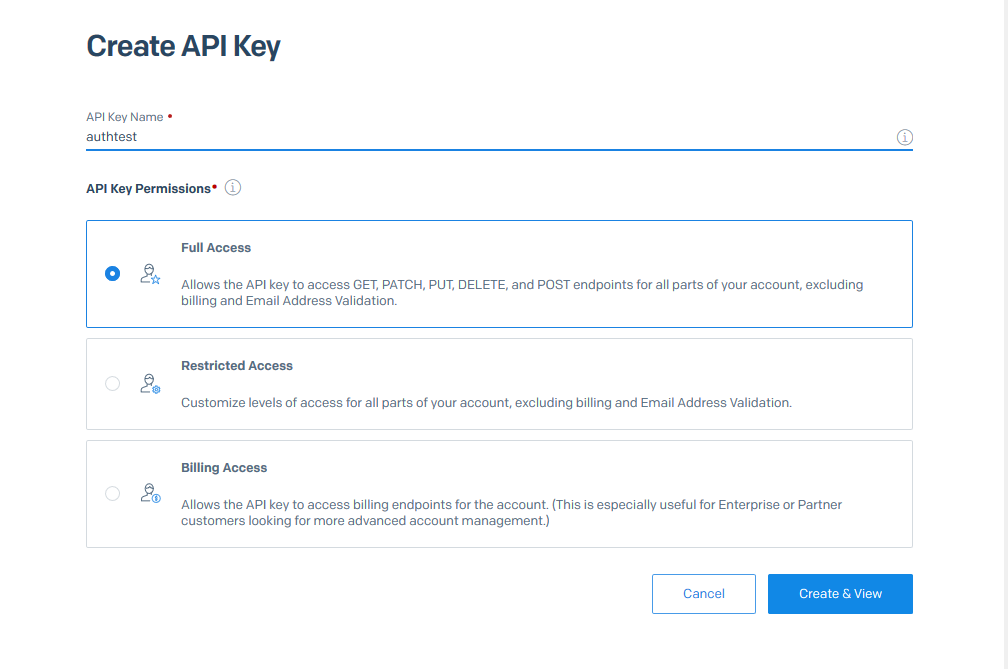
Skapa ett konto i <https://signup.sendgrid.com/>

Det är en mailserver. Man måste använda någon form av mailserver för att kunna skicka e-post. Man kan även använda gmail server eller AWS ses också.

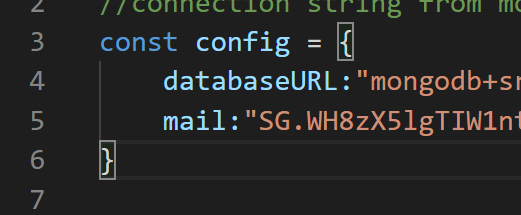
Efter man har bekräfta mailet ska man skapa ett api nyckel:

Skapa en api key:





Lägg nyckeln i : config.js filen



och importera den i router filen där du ska använda sendgrid mailet:

const config = require("../config/config");

Installera : nodemailer och nodemailer-sendgrid-transport paketen :

npm i --save nodemailer nodemailer-sendgrid-transport

\*Skapa en transporter med hjälp av sendgrid api key: (Man kunde använda något annat mail server istället för sendGridTransport för att kunna skapa transport (createTransport))

const transporter = nodemailer.createTransport(sendGridtransport({

auth:{

api\_key:config.mail

}

}))

i din skapa user konto/profil router:

router.post('/', async (req, res) => {

//joi valideringen

const { error } = validateUser(req.body);

if (error) return res.status(400).send(error.details[0].message);

//kolla upp om användare finns redan

let user = await User.findOne({ email: req.body.email });

if (user) return res.status(400).send('User already registered.');

var salt = await bcrypt.genSalt(10);

var hash = await bcrypt.hash(req.body.password, salt);

//skapar nya användare

user = new User({

username: req.body.username,

email:req.body.email,

password:hash

})

await user.save()

transporter.sendMail({

to:user.email,

from: "dinbusinessmail@com.se",

subject:"Login succeeded",

html: "<h1> Välkommen... </h1>"

})

//skicka nya skapande användare till client

res.render("userProfile", {user})

});

From och med nu om du skapar ett konto kommer du få ett mail.

Reset password:

Lägg till två fält i User modellen:

resetToken:String,

resetTokenExpiration: Date,

Skapa en router för reset get och post:

Importera node egna crypto paket.

const crypto = require("crypto");

router.get("/reset", (req, res)=>{

res.render("reset")

})

router.post("/reset", (req, res, next)=>{

crypto.randomBytes(32, (err, token)=>{

if(err){

res.redirect("/reset");

}

//resettoken måste sparas i databasen

const resetToken = token.toString("hex");

User.findOne({email:req.body.resetEmail}).then(async(user)=>{

if(!user) return res.redirect("/signin");

user.resetToken= resetToken;

user.resetTokenExpiration = Date.now() + 1000000;

console.log(user)

await user.save();

})

.then((result)=>{

transporter.sendMail({

to:req.body.resetEmail,

from: "dinbusinessmail@com.se",

subject:"Password reset",

html: `Your reset link <a href="http://localhost:8000/reset/${resetToken}">

link </a>`

})

res.redirect("/")

})

.catch(err=>console.log(err))

})

})

Skapa en reset view :

<%- include("partial/header.ejs") %>

<form action="/reset" method="POST">

<input type="email" name="resetEmail" id="">

<button type="submit">send</button>

</form>

Skapa en router för att fånga : "[http://localhost:8000/reset/${resetToken](http://localhost:8000/reset/$%7BresetToken)} länken:

router.get("/reset/:token", (req, res)=>{

res.render("resetForm")

})

Skapa en resetForm.ejs:

<form action="/resetForm" method="post">

<input type="password" name="" id="">

<input type="password" placeholder="upprepa lösenordet" name="" id="">

<input type="submit" value="Ändra">

</form>

Skriva om router: (Säkert sätt; bara änvändare med resetToken kommer att ha tillgång till reset formuläret

router.get("/reset/:token", (req, res)=>{

const token = req.params.token;

User.findOne({resetToken:token, resetTokenExpiration: {$gt: Date.now()}}).then( (user)=>{

res.render("resetForm", {userId: user.\_id, passwordToken: user.resetToken}})

})

})

Skriva om resetForm.ejs så att du får tillgång till userId och passwordToken:

<form action="/resetForm" method="post">

<input type="password" name="password" id="">

<input type="password" placeholder="upprepa lösenordet" name="" id="">

<input type="hidden" name="userId" id="" value="<%=userId%>">

<input type="hidden" name="passwordToken" value="<%=passwordToken%>">

<input type="submit" value="Ändra">

</form>

ResetForm post router:

router.post("/resetForm", (req, res) => {

const password = req.body.password;

const userId = req.body.userId;

const passwordToken = req.body.passwordToken;

let resetUser;

User

.findOne({

resetToken: passwordToken.toString(),

resetTokenExpiration: { $gt: Date.now()} ,

\_id:userId}

).then((user)=>{

resetUser =user;

return bcrypt.hash(password, 12);

})

.then((hashedPassword)=>{

resetUser.password = hashedPassword;

resetUser.resetToken = undefined;

resetUser.resetTokenExpiration = undefined;

resetUser.save();

return res.redirect("/signIn")

})

})