Implementando a Recuperação de Senha

Objetivos da Aula:

- Implementar a funcionalidade de recuperação de senha.
- Enviar e-mails com links para redefinição de senha.
- Configurar rotas e controladores para o processo de redefinição de senha.

1: Verifique o Banco de Dados:

Certifique-se de que a tabela **users** possui as colunas **reset_password_token** e **reset_password_expires**. Elas são necessárias para armazenar o token de redefinição de senha e o tempo de expiração.

- Você pode verificar isso executando o seguinte comando no MySQL:

Código SQL

```
DESCRIBE users;
```

- Se essas colunas não existirem, adicione-as:

Código SQL

```
ALTER TABLE users

ADD COLUMN reset_password_token VARCHAR(255) NULL,

ADD COLUMN reset_password_expires DATETIME NULL;
```

2: Variáveis de Ambiente:

- As variáveis de ambiente são usadas para armazenar informações sensíveis, como credenciais de e-mail. Elas são definidas no arquivo **.env** na raiz do projeto.
- Adicione as seguintes variáveis ao arquivo **.env**:

```
EMAIL_USER=seuemail@gmail.com
EMAIL_PASS=suaSenhaDeEmail
```

3: Instalando Dependências Necessárias

Para implementar a recuperação de senha, precisaremos de uma biblioteca para enviar e-mails. Vamos usar **nodemailer** para isso.

Ação: Execute o seguinte comando para instalar **nodemailer**:

```
npm install nodemailer
```

4: Configurando o Serviço de E-mail

Agora que temos **nodemailer** instalado, vamos configurar o serviço de envio de e-mails.

Ação:

1. Crie a pasta **services** na raiz do projeto (se ainda não existir):
2. Dentro da pasta **services,** crie o arquivo **emailService.js:**
3. Adicione o seguinte código ao **emailService.js:**

Código JavaScript

```
const nodemailer = require('nodemailer');

// Configuração do serviço de e-mail
const transporter = nodemailer.createTransport({
  service: 'gmail', // Use o serviço de e-mail de sua escolha
  auth: {
    user: process.env.EMAIL_USER, // Seu e-mail definido nas variáveis de ambiente
    pass: process.env.EMAIL_PASS  // Sua senha de e-mail definida nas variáveis de ambiente
  }
});

// Função para enviar e-mail
const sendEmail = (to, subject, text) => {
  const mailOptions = {
    from: process.env.EMAIL_USER,
    to,
    subject,
    text
  };

  transporter.sendMail(mailOptions, (error, info) => {
    if (error) {
      return console.log('Erro ao enviar e-mail:', error);
    }
    console.log('E-mail enviado:', info.response);
  });
};

module.exports = { sendEmail };
```

Explicação do Código:

- **nodemailer.createTransport:** Configura o transporte de e-mail usando o serviço (Gmail, no exemplo) e as credenciais definidas nas variáveis de ambiente.
- **sendEmail:** Função que recebe o destinatário, assunto e texto do e-mail e utiliza transporter.sendMail para enviar o e-mail.

5: Criando Rotas e Controladores para Recuperação de Senha

Vamos criar as rotas e controladores responsáveis por solicitar a redefinição de senha e realizar a redefinição.

Ação:

1. **Crie as rotas para recuperação de senha:**

Dentro da pasta **routes**, crie o arquivo **auth.js** (se ainda não existir

   -):
2. **Adicione o seguinte código ao auth.js:**

Código JavaScript

```
const express = require('express');
const router = express.Router();
const authController = require('../controllers/authController');

// Rota para solicitar redefinição de senha
router.post('/request-password-reset', authController.requestPasswordReset);

// Rota para redefinir a senha
router.post('/reset-password', authController.resetPassword);

module.exports = router;
```

Explicação do Código:

- **/request-password-reset:** Esta rota permite que o usuário solicite um link de redefinição de senha, que será enviado por e-mail.
- **/reset-password:** Esta rota permite ao usuário redefinir sua senha usando o token recebido por e-mail.

3. **Crie as funções no controlador:**

Dentro da pasta **controllers**, crie ou edite o arquivo **authController.js**:

```
touch controllers/authController.js
```

4. **Adicione ou atualize o seguinte código em authController.js:**

Código JavaScript

```
const crypto = require('crypto');
const db = require('../config/db');
const bcrypt = require('bcrypt');
const sendEmail = require('../services/emailService').sendEmail;

// Função para solicitar redefinição de senha
```

```
const requestPasswordReset = async (req, res) => {
  const { email } = req.body;

  try {
    const [user] = await db.promise().query('SELECT * FROM users WHERE email = ?', [email]);

    if (user.length === 0) {
      return res.status(404).send('Usuário não encontrado');
    }

    const token = crypto.randomBytes(20).toString('hex'); // Gera um token aleatório
    const expireDate = new Date(Date.now() + 3600000); // 1 hora para expiração

    await db.promise().query('UPDATE users SET reset_password_token = ?,
reset_password_expires = ? WHERE email = ?', [token, expireDate, email]);

    const resetLink = `http://localhost:3000/reset-password/${token}`; // Link para redefinição
de senha
    sendEmail(email, 'Recuperação de Senha', `Por favor, clique no link para redefinir sua senha:
${resetLink}`);

    res.send('E-mail de recuperação de senha enviado');
  } catch (err) {
    console.error('Erro ao solicitar redefinição de senha:', err);
    res.status(500).send('Erro ao solicitar redefinição de senha');
  }
};

// Função para redefinir a senha
const resetPassword = async (req, res) => {
  const { token, newPassword } = req.body;

  try {
    const [user] = await db.promise().query('SELECT * FROM users WHERE
reset_password_token = ? AND reset_password_expires > NOW()', [token]);

    if (user.length === 0) {
      return res.status(400).send('Token inválido ou expirado');
    }

    const hashedPassword = await bcrypt.hash(newPassword, 10); // Criptografa a nova senha

    await db.promise().query('UPDATE users SET password = ?, reset_password_token = NULL,
reset_password_expires = NULL WHERE id = ?', [hashedPassword, user[0].id]);

    res.send('Senha redefinida com sucesso');
  } catch (err) {
    console.error('Erro ao redefinir senha:', err);
    res.status(500).send('Erro ao redefinir senha');
  }
};
```

module.exports = { requestPasswordReset, resetPassword };

Explicação do Código:

- **requestPasswordReset:** Gera um token de redefinição de senha e envia um e-mail ao usuário com um link para redefinir a senha.
- **resetPassword:** Redefine a senha do usuário após verificar a validade do token.

7: Configurando o Servidor para Usar as Novas Rotas

Finalmente, vamos garantir que o servidor está configurado para usar essas novas rotas.

Ação:

1. **Atualize o arquivo server.js para incluir as rotas de autenticação:**

Código JavaScript

```
const dotenv = require('dotenv'); // Importa o pacote dotenv para gerenciar variáveis de ambiente

// Carregar as Variáveis de Ambiente
dotenv.config(); // Carrega as variáveis definidas no arquivo .env para process.env

// Importar as Bibliotecas
const express = require('express'); // Importa o framework Express
const cors = require('cors'); // Importa o pacote cors para permitir requisições de diferentes origens
const bodyParser = require('body-parser'); // Importa o pacote body-parser para analisar o corpo das requisições HTTP

const db = require('./config/db'); // Importa a conexão com o banco de dados

// Inicializar nova aplicação Express
const app = express(); // Inicializa uma nova aplicação Express

// Configurar o CORS e o body-parser
app.use(cors()); // Habilita o CORS para todas as rotas
app.use(bodyParser.json()); // Configura o body-parser para analisar requisições JSON

// Importar as rotas de transações e autenticação
const transactionsRoutes = require('./routes/transactions'); // Importa as rotas de transações
const authRoutes = require('./routes/auth'); // Importa as rotas de autenticação

// Usar as rotas de transações e autenticação para as requisições
app.use('/api/transactions', transactionsRoutes); // Configura o servidor para usar as rotas de transações
app.use('/api/auth', authRoutes); // Configura o servidor para usar as rotas de autenticação

// Rota inicial para testar o servidor
app.get('/', (req, res) => {
```

```
  res.send('Servidor está rodando'); // Define uma rota inicial para testar o servidor
});

// Configurar o servidor para uma porta específica
const PORT = process.env.PORT || 3000; // Define a porta a partir da variável de ambiente ou usa a porta 3000 como padrão
app.listen(PORT, () => {
  console.log(`Servidor rodando na porta ${PORT}`); // Loga uma mensagem informando que o servidor está rodando
});
```

Parte Prática (1 hora):

Ação:

1. **Solicitar a redefinição de senha:**
   - Método: POST
   - URL: http://localhost:3000/api/auth/request-password-reset
   - Corpo da requisição (JSON):

     Código JSON

     ```
     {
       "email": "joanadark@apifinance.com"
     }
     ```

2. **Redefinir a senha:**
   - Método: POST
   - URL: http://localhost:3000/api/auth/reset-password
   - Corpo da requisição (JSON):

     Código JSON

     ```
     {
       "token": "<token-recebido-no-email>",
       "newPassword": "novaSenha123"
     }
     ```

Testes Importantes:

- Verifique se o token é gerado e armazenado corretamente no banco de dados.
- Certifique-se de que o e-mail é enviado corretamente e contém o link correto para redefinição de senha.
- Teste a redefinição de senha para garantir que o token expira após o tempo definido (1 hora).