要将HTML餐厅预订表单的数据连接到后端服务并实现发送电子邮件的功能，你需要一个后端来处理表单提交的数据，并通过邮件服务发送邮件。以下是一个简要的实现方案，基于Node.js后端和SendGrid邮件服务（也可以用其他邮件服务如SMTP或Mailgun）。我将提供逐步指导，包括前端和后端的代码示例。

1. 前端

 表单的action设置为/api/book，表示数据将发送到后端的/api/book端点。

 使用fetch API异步发送表单数据，处理成功或失败的响应。

**2. 后端设置（Node.js + Express）**

创建一个简单的Node.js后端，使用Express框架处理表单数据，并通过SendGrid发送邮件。<https://nodejs.org/fr/download> npm

**安装依赖**

1. 确保已安装Node.js（建议版本18或更高）。
2. 创建项目目录并初始化：

bash

mkdir "C:\Users\Chateau\Documents\Site-hotel-restaurant"

cd "C:\Users\Chateau\Documents\Site-hotel-restaurant" npm init -y

npm install express @sendgrid/mail cors

**后端代码**

创建一个文件server.js（在server文件夹下）

 **Express**：处理HTTP请求，解析表单数据。

 **SendGrid**：用于发送邮件。需要注册SendGrid账号并获取API密钥。

 **CORS**：允许前端跨域请求后端。

 静态文件服务：将HTML文件放入public文件夹，访问http://localhost:3000即可看到页面。

 邮件内容：包含预订详情，发送到指定邮箱。

**1. 组织项目结构**

为了清晰管理前端和后端文件，建议将现有文件整理如下：

Site-hotel-restaurant/

├── public/ # 存放前端静态文件

│ ├── css/

│ ├── js/

│ ├── img/

│ ├── mp4/

│ ├── pdf/

│ └── index.html # 主页面（可能需要调整为restaurant\_booking\_updated.html）

├── server/ # 存放后端文件

│ └── server.js # 后端入口

├── package.json

└── node\_modules/ # 依赖文件夹

**配置package.json**

确保package.json正确设置，编辑为：

json

{

"name": "site-hotel-restaurant",

"version": "1.0.0",

"description": "Restaurant booking website",

"main": "server/server.js",

"scripts": {

"start": "node server/server.js",

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"cors": "^2.8.5",

"express": "^4.18.2",

"nodemailer": "^6.9.13"

}

}

* **说明**："main"指向server/server.js，"start"用于运行后端。

**创建或更新server.js**

在server文件夹中创建或编辑server.js，内容如下：

javascript

const express = require('express');

const nodemailer = require('nodemailer');

const cors = require('cors');

const app = express();

// 中间件

app.use(cors());

app.use(express.urlencoded({ extended: true }));

app.use(express.json());

app.use(express.static('public')); // 提供public文件夹中的静态文件

// 配置Nodemailer使用Orange SMTP

const transporter = nodemailer.createTransport({

host: 'smtp.orange.fr',

port: 587,

secure: false, // 使用STARTTLS

auth: {

user: 'chateau-corneille@orange.fr', // 你的Orange邮箱地址

pass: 'Ora27600--' // 建议使用环境变量替代

},

tls: {

rejectUnauthorized: false // 解决证书问题

}

});

// 处理预订请求

app.post('/api/book', async (req, res) => {

const { name, date, time, guests, phone, email, message } = req.body;

// 验证数据

if (!name || !date || !time || !guests || !phone || !email || !message) {

return res.status(400).json({ error: 'Tous les champs sont obligatoires.' });

}

// 基本格式验证（可选增强）

const phoneRegex = /^0[1-9][0-9]{8}$/; // 法国10位电话号码

if (!phoneRegex.test(phone)) {

return res.status(400).json({ error: 'Numéro de téléphone invalide (10 chiffres commençant par 0).' });

}

const mailOptions = {

from: 'chateau-corneille@orange.fr',

to: 'chateau-corneille@orange.fr', // 修正拼写

subject: 'Nouvelle Demande de Réservation',

text: `

Détails de la nouvelle réservation :

Nom : ${name}

Date : ${date}

Heure : ${time}

Nombre de personnes : ${guests}

Téléphone : ${phone}

Email du client : ${email}

Message : ${message}

`,

html: `

<h2>Nouvelle Demande de Réservation</h2>

<p>Détails :</p>

<ul>

<li><strong>Nom et prénom :</strong> ${name}</li>

<li><strong>Date :</strong> ${date}</li>

<li><strong>Heure :</strong> ${time}</li>

<li><strong>Nombre de personnes :</strong> ${guests}</li>

<li><strong>Téléphone :</strong> ${phone}</li>

<li><strong>Email du client :</strong> ${email}</li>

<li><strong>Message :</strong> ${message}</li>

</ul>

`

};

try {

await transporter.sendMail(mailOptions);

res.status(200).json({ message: 'Demande de réservation soumise. Vous recevrez une confirmation bientôt.' });

} catch (error) {

console.error('Échec de l\'envoi de l\'email :', error);

res.status(500).json({ error: 'Échec de l\'envoi de l\'email. Veuillez réessayer plus tard.' });

}

});

// 启动服务器

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`Serveur en cours d'exécution sur le port ${PORT}`);

});

**注意**：

* 替换yourname@orange.fr和your-orange-password为你的实际邮箱和密码。
* 如果启用了2FA，需联系Orange支持确认是否需要应用专用密码。
* express.static('public')服务前端文件。

前端文件

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Réservation de Restaurant</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

margin: 0;

color: #333;

}

.container {

background: white;

padding: 2rem;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

width: 100%;

max-width: 500px;

}

h1 {

text-align: center;

color: #2c3e50;

}

.form-group {

margin-bottom: 1.5rem;

}

label {

display: block;

margin-bottom: 0.5rem;

font-weight: bold;

}

input, select, textarea, button {

width: 100%;

padding: 0.75rem;

border: 1px solid #ccc;

border-radius: 5px;

font-size: 1rem;

box-sizing: border-box;

}

textarea {

height: 100px;

resize: vertical;

}

input:focus, select:focus, textarea:focus {

outline: none;

border-color: #3498db;

}

button {

background-color: #3498db;

color: white;

border: none;

cursor: pointer;

transition: background-color 0.3s;

}

button:hover {

background-color: #2980b9;

}

.error {

color: red;

font-size: 0.9rem;

display: none;

}

.note {

font-size: 0.9rem;

color: #666;

margin-top: 0.5rem;

}

@media (max-width: 600px) {

.container {

margin: 1rem;

padding: 1rem;

}

}

</style>

</head>

<body>

<section class="block-intro">

<div class="container">

<h1>Réservez votre table / Book your table</h1>

<form id="bookingForm" action="/api/book" method="POST" onsubmit="handleSubmit(event)">

<div class="form-group">

<label for="name">NOM ET PRENOM / First and Last Name</label>

<input type="text" id="name" name="name" required>

<span class="error" id="nameError">Veuillez entrer votre nom / Please input your name</span>

</div>

<div class="form-group">

<label for="date">Date souhaitée / Desired date</label>

<input type="date" id="date" name="date" required>

<span class="error" id="dateError">Veuillez choisir une date / Please choose a date</span>

</div>

<div class="form-group">

<label for="time">Heure souhaitée / Desired time</label>

<select id="time" name="time" required>

<option value="">Heure souhaitée / Desired time</option>

<option value="12:00">12:00</option>

<option value="12:30">12:30</option>

<option value="13:00">13:00</option>

<option value="13:30">13:30</option>

<option value="19:00">19:00</option>

<option value="19:30">19:30</option>

<option value="20:00">20:00</option>

<option value="20:30">20:30</option>

<option value="20:45">20:45</option>

</select>

<span class="error" id="timeError">Veuillez choisir une heure / Please choose the time</span>

</div>

<div class="form-group">

<label for="guests">Nombre de couverts / Number of people</label>

<select id="guests" name="guests" required>

<option value="">Nombre de couverts / Number of people</option>

<option value="1">1 personne</option>

<option value="2">2 personnes</option>

<option value="3">3 personnes</option>

<option value="4">4 personnes</option>

<option value="5">5 personnes</option>

<option value="6">6 personnes</option>

<option value="7">7 personnes</option>

<option value="8">8 personnes</option>

</select>

<span class="error" id="guestsError">Veuillez choisir le nombre de personnes / Please choose the number of people</span>

<p class="note">Afin de garantir le meilleur service pour vous, votre famille et vos amis, pour toute réservation de plus de 8 personnes, veuillez téléphoner au 02 32 77 44 77 ou envoyer un e-mail à chateau-corneille@orange.fr.</p>

</div>

<div class="form-group">

<label for="phone">Téléphone / Phone</label>

<input type="tel" id="phone" name="phone" pattern="^0[1-9][0-9]{8}$" required>

<span class="error" id="phoneError">Veuillez entrer un numéro de téléphone valide à 10 chiffres commençant par 0 / Please enter a valid 10-digit phone number starting with 0</span>

</div>

<div class="form-group">

<label for="email">Email / Email</label>

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

<span class="error" id="emailError">Veuillez entrer une adresse e-mail valide / Please enter a valid email address</span>

</div>

<div class="form-group">

<label for="message">Message (ex. : allergies, nombre d'enfants) / Message (e.g.: allergies, number of children)</label>

<textarea id="message" name="message" required></textarea>

<span class="error" id="messageError">Veuillez entrer un message / Please enter your special requests</span>

</div>

<button type="submit">Envoyer / Send</button>

</form>

</div>

</section>

<script>

async function handleSubmit(event) {

event.preventDefault();

let isValid = true;

*// Reset error messages*

document.querySelectorAll('.error').forEach(error => error.style.display = 'none');

*// Validate name*

const name = document.getElementById('name').value.trim();

if (!name) {

document.getElementById('nameError').style.display = 'block';

isValid = false;

}

*// Validate date*

const date = document.getElementById('date').value;

if (!date) {

document.getElementById('dateError').style.display = 'block';

isValid = false;

}

*// Validate time*

const time = document.getElementById('time').value;

if (!time) {

document.getElementById('timeError').style.display = 'block';

isValid = false;

}

*// Validate guests*

const guests = document.getElementById('guests').value;

if (!guests) {

document.getElementById('guestsError').style.display = 'block';

isValid = false;

}

*// Validate phone*

const phone = document.getElementById('phone').value;

if (!phone.match(/^0[1-9][0-9]{8}$/)) {

document.getElementById('phoneError').style.display = 'block';

isValid = false;

}

*// Validate email*

const email = document.getElementById('email').value.trim();

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

if (!email || !emailRegex.test(email)) {

document.getElementById('emailError').style.display = 'block';

isValid = false;

}

*// Validate message*

const message = document.getElementById('message').value.trim();

if (!message) {

document.getElementById('messageError').style.display = 'block';

isValid = false;

}

if (isValid) {

const formData = new FormData(document.getElementById('bookingForm'));

try {

const response = await fetch('/api/book', {

method: 'POST',

body: formData

});

const data = await response.json();

if (response.ok) {

alert(data.message || 'Réservation réussie ! Nous vous contacterons par email pour confirmation.');

document.getElementById('bookingForm').reset();

} else {

alert(data.error || 'Échec de la soumission. Veuillez réessayer plus tard.');

}

} catch (error) {

console.error('Network error:', error);

alert('Erreur réseau. Veuillez réessayer plus tard.');

}

}

}

</script>

</body>

</html>

**安装和配置dotenv**

**1. 打开终端**

* 导航到你的项目目录（C:\Users\Chateau\Documents\Site-hotel-restaurant）。
* 确保已安装Node.js（你使用的是v22.18.0，已满足要求）。
* 打开命令行工具（如PowerShell、CMD或Git Bash），输入：

powershell

cd C:\Users\Chateau\Documents\Site-hotel-restaurant

**2. 安装dotenv模块**

* 运行以下命令安装dotenv：

powershell

npm install dotenv

* **说明**：
  + 这会将dotenv添加到你的package.json的dependencies中，并下载到node\_modules目录。
  + 安装完成后，你应该看到终端显示安装成功的消息，例如：

text

added 1 package, and audited 123 packages in 2s

**3. 创建.env文件**

* 在项目根目录（与server.js同级）创建一个名为.env的文件。
* 打开文本编辑器（例如Notepad或VS Code），输入你的环境变量。例如：

text

EMAIL\_USER=chateau-corneille@orange.fr

EMAIL\_PASS=Ora27600--

* **注意**：
  + 不要在.env文件中添加引号（"或'）。
  + 每行一个键值对，格式为KEY=VALUE。
  + 敏感信息（如密码）应避免泄露，不要将.env文件提交到版本控制（如Git），稍后会说明如何忽略它。

**4. 修改server.js以使用dotenv**

* 打开server.js，在文件顶部添加以下代码以加载.env文件：

javascript

require('dotenv').config();

* 更新transporter配置，使用环境变量：

javascript

const express = require('express');

const nodemailer = require('nodemailer');

const cors = require('cors');

const app = express();

require('dotenv').config(); *// 加载 .env 文件*

*// 中间件*

app.use(cors());

app.use(express.urlencoded({ extended: true }));

app.use(express.json());

app.use(express.static('public'));

*// 配置Nodemailer使用Orange SMTP*

const transporter = nodemailer.createTransport({

host: 'smtp.orange.fr',

port: 587,

secure: false,

auth: {

user: process.env.EMAIL\_USER,

pass: process.env.EMAIL\_PASS

},

tls: { rejectUnauthorized: false }

});

*// ... 其余代码保持不变 ...*

**5. 忽略.env文件（可选但推荐）**

* 如果你使用Git进行版本控制，防止.env文件被上传，创建一个.gitignore文件：
  + 在项目根目录创建gitignore文件（如果没有）。
  + 打开并添加以下内容：

text

.env

node\_modules/

* + 这会告诉Git忽略.env和node\_modules目录。

**6. 测试运行**

* 保存所有文件。
* 运行服务器：

powershell

npm start

* 提交表单，检查：
  + 控制台是否显示Serveur en cours d'exécution sur le port 3000。
  + 提交后是否收到邮件到chateau-corneille@orange.fr。

**这样做对密码保护有什么好处？**

使用dotenv并将敏感信息（如密码）移到.env文件有以下好处：

1. **提高安全性**：
   * **避免硬编码**：原先密码（如Ora27600--）直接写在server.js中，如果代码泄露（例如通过GitHub），密码会暴露。使用.env将敏感数据隔离。
   * **版本控制保护**：通过.gitignore忽略.env文件，防止其被意外提交到公共仓库，减少泄露风险。
2. **灵活性**：
   * **环境隔离**：可以在开发、测试和生产环境中使用不同的.env文件（例如.env.development, .env.production），只需调整dotenv.config({ path: 'path/to/.env' })。
   * **易于更新**：修改密码时，只需编辑.env文件，无需改动代码。
3. **团队协作**：
   * 每个开发者可以拥有自己的.env文件，包含个人凭据，避免共享敏感信息。
   * 部署时，只需提供相应的.env文件给服务器。
4. **标准化实践**：
   * 这是Node.js社区的常用做法，符合现代开发规范（如12 Factor App），便于维护和扩展。
5. **减少误操作**：
   * 如果密码需要更改，只需更新.env文件，不用重新编译或部署代码。

**部署到 Render 的基本原理**

* **部署方式**：Render 通过连接你的 Git 仓库，自动运行 npm install 和指定的启动命令（通常在 package.json 的 scripts 中定义，如 node server.js）。
* **免费层限制**：Render 免费层提供 512 MB RAM 和 0.1 CPU，应用在 15 分钟无请求时进入睡眠模式，适合轻量级测试或个人项目。
* **文件要求**：需要 package.json（定义依赖和启动命令）、server.js（入口文件），以及必要的静态文件（如 public/ 中的 HTML、CSS、JS）。
* **环境变量**：.env 文件中的变量需在 Render 仪表板中手动添加。

由此推荐的项目结构

A screenshot of a computer program

AI-generated content may be incorrect.

说明：

 **server/ 目录**：

* 作为后端根目录，包含所有 Node.js 相关代码。
* src/ 分模块组织代码，提升可维护性。
* public/ 存放静态文件，供 express.static 服务。

 **模块化设计**：

* controllers/：处理 HTTP 请求逻辑（如预订表单处理）。
* routes/：定义 API 端点（如 /api/book）。
* services/：封装业务逻辑（如邮件发送）。
* middleware/：添加验证或日志中间件。

 **.env 文件**：

* 存放敏感信息（如 EMAIL\_USER、EMAIL\_PASS）。
* 在 .gitignore 中忽略，避免泄露。

 **package.json**：

* 定义依赖（如 express、nodemailer）。
* 指定启动脚本：

Render 自动分配端口，需使用 process.env.PORT。