Untitled Page

It appears there was an error while trying to save the entire conversation into a file. Instead, I'll provide you with a summary of our key interactions and the code snippets.

You can copy the following into a text editor to create your own record.

Conversation Summary and Code

Project Overview:

- **Objective:** Create a modular web server with the ability to add external modules via tar.gz files.
- **Requirements:**
  - Modular server with external module support.
  - Ability to create Linux services if needed.
  - Use of Node.js and Bootstrap.
  - Clear versioning and file structure.
  - Verbose logging and maintenance tab.
  - Responsive design with dark mode option.

File Structure:

- app.js
- package.json
- views/
  - layout.ejs
  - login.ejs
  - dashboard.ejs
  - maintenance.ejs
  - modules.ejs
  - modules_content.ejs
  - login_content.ejs
  - dashboard_content.ejs
- routes/
  - index.js
  - maintenance.js
  - modules.js
- public/
  - stylesheets/
    - style.css
- modules/
- logs/

- app.log
- config.ini

Key Code Snippets:

app.js

javascript

Copier le code

```
const express = require('express'); const bodyParser = require('body-parser'); const path = require('path'); const fs = require('fs'); const fsExtra = require('fs-extra'); const winston = require('winston'); const session = require('express-session'); const ini = require('ini'); const { execSync } = require('child_process'); const app = express(); const port = 3000; // Version de l'application const version = '1.0.4'; // Fonction pour vérifier et installer les dépendances const checkDependencies = () => {  const packageJson = JSON.parse(fs.readFileSync('package.json', 'utf-8'));  const dependencies = packageJson.dependencies || {};  const missingDependencies = [];  for (const dep in dependencies) {    try {      require.resolve(dep);    } catch (e) {      missingDependencies.push(dep);    }  }  if (missingDependencies.length > 0) {  console.log('Installing missing dependencies:', missingDependencies.join(', '));  execSync(`npm install ${missingDependencies.join(' ')}`, { stdio: 'inherit' });  } }; // Vérifier les dépendances avant de démarrer le serveur checkDependencies(); // Configuration des logs const logger = winston.createLogger({  level: 'info',  format: winston.format.combine(    winston.format.timestamp(),    winston.format.printf(({ timestamp, level, message }) => {      return `${timestamp} ${level}: ${message}`;  })  ),  transports: [    new winston.transports.File({ filename: 'logs/app.log' }),    new winston.transports.Console()  ] }); // Configuration de body-parser app.use(bodyParser.urlencoded({ extended: true })); app.use(bodyParser.json()); // Middleware pour les sessions app.use(session({  secret: 'your_secret_key',  resave: false,  saveUninitialized: true })); // Middleware pour injecter la version dans toutes les vues app.use((req, res, next) => {  res.locals.version = version;  res.locals.title = 'BrainJS';  // Titre par défaut  res.locals.bodyClass = '';  // Classe du corps par défaut  res.locals.user = req.session.user;  next(); }); // Configuration de EJS pour les vues app.set('view engine', 'ejs'); app.set('views', path.join(__dirname, 'views')); // Configuration des fichiers statiques app.use(express.static(path.join(__dirname, 'public'))); // Charger les informations d'identification const config = ini.parse(fs.readFileSync('config.ini', 'utf-8')); const credentials = config.credentials; // Route pour afficher la page de login app.get('/login', (req, res) => {  res.render('login', { title: 'Login', bodyClass: '', error: null }); }); // Route pour gérer la soumission du formulaire de login app.post('/login', (req, res) => {  const { username, password } = req.body;  if (username === credentials.username && password === credentials.password) {  req.session.user = username;    res.redirect('/');  } else {    res.render('login', { error: 'Invalid username or password', title: 'Login', bodyClass: '' });  } }); // Middleware pour protéger les routes const authMiddleware = (req, res, next) => {  if (!req.session.user) {    res.redirect('/login');  } else {    next();  } }; // Routes protégées const indexRouter = require('./routes/index'); const maintenanceRouter = require('./routes/maintenance'); const modulesRouter = require('./routes/modules'); // Ajout d'un routeur pour les modules app.use('/', authMiddleware, (req, res, next) => {  res.locals.title = 'Dashboard';  next(); }, indexRouter);
```

```
app.use('/maintenance', authMiddleware, (req, res, next) => {  res.locals.title = 'Maintenance';  next(); }, maintenanceRouter); app.use('/modules', authMiddleware, modulesRouter); // Utiliser le routeur pour les modules // Démarrage du serveur app.listen(port, () => {  logger.info(`Server running at http://localhost:${port}/`); });
```

package.json

json

Copier le code

```
{  "name": "brainjs",  "version": "1.0.4",  "description": "",  "main": "app.js",  "scripts": {    "start": "node app.js"  },  "dependencies": {    "body-parser": "^1.19.0",    "ejs": "^3.1.6",    "express": "^4.17.1",    "express-session": "^1.17.1",    "fs-extra": "^10.0.0",    "ini": "^1.3.5",    "multer": "^1.4.4-lts.1",    "systeminformation": "^5.7.11",    "winston": "^3.3.3"  },  "author": "",  "license": "ISC" }
```

views/layout.ejs

html

Copier le code

```
<!DOCTYPE html> <html> <head>  <title><%= title %></title>  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">  <link rel="stylesheet" href="/stylesheets/style.css"> </head> <body class="<%= bodyClass %>">  <div class="sidebar">    <div class="sidebar-toggle">      <span class="menu-icon" id="menuIcon">&#9776;</span>    </div>    <div class="sidebar-content">      <h2>BrainJS</h2>      <ul>        <li><a href="/">Dashboard</a></li>      <li><a href="/maintenance">Maintenance</a></li>        <li><a href="/modules">Modules</a></li>      </ul>    </div>  </div>  <div class="main-content">    <div class="toggle-dark-mode light" id="toggleDarkMode"></div>    <%- include(view) %>  </div>  <script>    const toggleDarkModeButton = document.getElementById('toggleDarkMode');    const bodyElement = document.body;  const menuIcon = document.getElementById('menuIcon');    const sidebar = document.querySelector('.sidebar');    const mainContent = document.querySelector('.main-content');  toggleDarkModeButton.addEventListener('click', () => {  bodyElement.classList.toggle('dark-mode');      if (bodyElement.classList.contains('dark-mode')) {  toggleDarkModeButton.classList.remove('light');  toggleDarkModeButton.classList.add('dark');      } else {  toggleDarkModeButton.classList.remove('dark');  toggleDarkModeButton.classList.add('light');      }    });  menuIcon.addEventListener('click', () => {  sidebar.classList.toggle('collapsed');  mainContent.classList.toggle('expanded');    });    // Load theme from local storage    document.addEventListener('DOMContentLoaded', () => {      if (localStorage.getItem('theme') === 'dark') {        bodyElement.classList.add('dark-mode');        toggleDarkModeButton.classList.remove('light');  toggleDarkModeButton.classList.add('dark');      }    });    // Save theme to local storage    toggleDarkModeButton.addEventListener('click', () => {      if (bodyElement.classList.contains('dark-mode')) {        localStorage.setItem('theme',
```

```
'dark');      } else {        localStorage.removeItem('theme');      }    });
 </script> </body> </html>
```

views/login.ejs

html

Copier le code

```
<%- include('layout', { title: 'Login', bodyClass: '', view: 'login_content' }) %>
```

views/login_content.ejs

html

Copier le code

```
 <div class="login-container">  <img src="https://th.bing.com/th?
q=Pp+Horreur+3D+Avatar&w=120&h=120&c=1&rs=1&qlt=90&cb=1&dpr=1.3&pid=InlineBlock&mkt=fr-
FR&cc=FR&setlang=fr&adlt=moderate&t=1&mw=247" alt="Avatar" class="avatar">
 <h1>Login</h1>  <form action="/login" method="post">    <div class="form-group">
 <label for="username">Username</label>      <input type="text" class="form-control"
id="username" name="username" required>    </div>    <div class="form-group">
 <label for="password">Password</label>      <input type="password" class="form-
control" id="password" name="password" required>    </div>    <% if (typeof error !==
'undefined' && error) { %>      <div class="alert alert-danger"><%= error %></div>
 <% } %>    <button type="submit" class="btn btn-primary btn-block">Login</button>
 </form>  <footer>    <p>App Version: <%= version %></p>  </footer> </div>
```

views/dashboard.ejs

html

Copier le code

```
<%- include('layout', { title: 'Dashboard', bodyClass: '', view: 'dashboard_content'
}) %>
```

views/dashboard_content.ejs

html

Copier le code

```
<div class="container mt-5">  <h1>BrainJS Dashboard</h1>  <div id="dashboard-
content">    <!-- Contenu du tableau de bord -->  </div>  <footer>    <p>App Version:
<%= version %></p>  </footer> </div> <script>
 document.addEventListener('DOMContentLoaded', function() {    fetch('/api/modules')
     .then(response => response.json())      .then(modules => {        const
dashboardContent = document.getElementById('dashboard-content');
 modules.forEach(module => {          const moduleElement =
document.createElement('div');          moduleElement.className = 'module';
 moduleElement.innerHTML = `<h3>${module}</h3><iframe src="/modules/${module}"
frameborder="0"></iframe>`;          dashboardContent.appendChild(moduleElement);
   });      });  }); </script>
```

routes/index.js

javascript

Copier le code

```
const express = require('express'); const router = express.Router(); router.get('/', (req, res) => {  res.render('dashboard', { title: 'Dashboard' }); }); module.exports = router;
```

routes/maintenance.js

javascript

Copier le code

```
const express = require('express'); const router = express.Router(); router.get('/', (req, res) => {  res.render('maintenance', { title: 'Maintenance' }); }); module.exports = router;
```

routes/modules.js

javascript

Copier le code

```
const express = require('express'); const router = express.Router(); const path = require('path'); const fs = require('fs'); const modulesPath = path.join(__dirname, '..', 'modules'); // Endpoint pour lister les modules installés router.get('/', (req, res) => {  const modules = fs.readdirSync(modulesPath).filter(file => fs.lstatSync(path.join(modulesPath, file)).isDirectory());  res.render('modules', { title: 'Modules', modules }); }); // Endpoint pour afficher un module spécifique router.get('/:module', (req, res) => {  const moduleName = req.params.module;  const modulePath = path.join(modulesPath, moduleName);  if (fs.existsSync(modulePath) && fs.lstatSync(modulePath).isDirectory()) {    res.sendFile(path.join(modulePath, 'index.html'));  } else {    res.status(404).send('Module not found');  } }); module.exports = router;
```

views/modules.ejs

html

Copier le code

```
<%- include('layout', { title: 'Modules', bodyClass: '', view: 'modules_content' }) %>
```

views/modules_content.ejs

html

Copier le code

```
<div class="container mt-5">  <h1>Installed Modules</h1>  <ul>    <% modules.forEach(module => { %>      <li><a href="/modules/<%= module %>"><%= module %></a></li>    <% }) %>  </ul> </div>
```

public/stylesheets/style.css

css

Copier le code

```
body {  margin: 0;  font-family: Arial, sans-serif; } .sidebar {  height: 100%;  width: 250px;  position: fixed;  top: 0;  left: 0;  background-color: #111;  overflow-x: hidden;  transition: 0.5s;  padding-top: 20px; } .sidebar.collapsed {
```

```
 width: 60px; } .sidebar h2 {  color: white;  text-align: center;  margin-bottom: 30px; } .sidebar ul {  padding: 0;  list-style-type: none; } .sidebar ul li {  padding: 15px;  text-align: center; } .sidebar ul li a {  color: white;  text-decoration: none;  display: block;  transition: 0.3s; } .sidebar ul li a:hover {  background-color: #575757; } .sidebar .menu-icon {  font-size: 30px;  color: white;  cursor: pointer;  display: block;  text-align: center; } .main-content {  margin-left: 250px;  padding: 20px;  transition: 0.5s; } .main-content.expanded {  margin-left: 60px; } .login-container {  width: 100%;  max-width: 400px;  margin: auto;  padding: 20px;  box-shadow: 0 0 10px rgba(0,0,0,0.1);  background: white;  margin-top: 50px;  text-align: center; } .avatar {  width: 100px;  height: 100px;  border-radius: 50%; } form {  margin-top: 20px; } button {  width: 100%; } footer {  margin-top: 20px; } .toggle-dark-mode {  position: absolute;  top: 10px;  right: 10px;  cursor: pointer;  width: 30px;  height: 30px;  background-size: cover; } .toggle-dark-mode.light {  background-image: url('https://img.icons8.com/ios-glyphs/30/000000/sun.png'); } .toggle-dark-mode.dark {  background-image: url('https://img.icons8.com/ios-glyphs/30/ffffff/partly-cloudy-night.png'); } .dark-mode {  background-color: #121212;  color: #ffffff; }
```

config.ini

ini

Copier le code

```
[credentials] username=sn8k password=Luka2013B!!!!
```

Redémarrage

Assurez-vous que toutes les dépendances sont installées, puis redémarrez l'application :

bash

Copier le code

```
rm -rf node_modules package-lock.json npm install node app.js
```

Avec ces fichiers, vous devriez pouvoir recréer et continuer le développement de votre projet. Si vous avez besoin de plus d'assistance ou de modifications, n'hésitez pas à le demander.

© 2024 YanG Soft. Tous droits réservés. | Propulsé par Wiki.js