Let's continue with the detailed exploration of the websockets/ directory in your NestJS project. This directory is pivotal for implementing real-time features using WebSockets, such as live dashboards and user notifications.

**📁 websockets/ — Real-Time Gateway Layer**

**📌 Purpose**

The websockets/ directory encapsulates all WebSocket gateways and related event handler. It facilitates real-time, bidirectional communication between the server and connected clients, enabling features like live updates, notifications, and interactive dashboard.

**🗂️ Suggested Directory Structur**



src/

└── websockets/

├── events/ # Event constants or handlers

│ ├── dashboard.events.ts

│ └── notification.events.ts

├── dashboard.gateway.ts # Real-time dashboard updates

├── notification.gateway.ts # Real-time user notifications

└── websocket.module.ts # Combines all gateways

``



- \*\*`events/`\*: Contains constants or handlers for various WebSocket evens.

- \*\*`dashboard.gateway.ts`\*: Handles real-time communications for dashboard updats.

- \*\*`notification.gateway.ts`\*: Manages real-time user notificatios.

- \*\*`websocket.module.ts`\*: Aggregates all WebSocket gateways into a single modue.

---

## 🧩 Implementation Details

### 1. \*\*Event Constants (`events/`)\*

Defining event constants ensures consistency across the application when emitting or listening to evets.

\*\*`dashboard.events.ts`\*



```typescript

// src/websockets/events/dashboard.events.ts

/\*\*

\* Event constants for dashboard-related WebSocket communications.

\*/

export const DASHBOARD\_EVENTS = {

UPDATE: 'dashboard\_update',

REFRESH: 'dashboard\_refresh',

};

``



\*\*`notification.events.ts`\*



```typescript

// src/websockets/events/notification.events.ts

/\*\*

\* Event constants for notification-related WebSocket communications.

\*/

export const NOTIFICATION\_EVENTS = {

NEW: 'notification\_new',

READ: 'notification\_read',

};

``



### 2. \*\*Dashboard Gateway (`dashboard.gateway.ts`)\*

This gateway manages real-time updates for the dashboard, broadcasting data to all connected cliets.



```typescript

// src/websockets/dashboard.gateway.ts

import {

WebSocketGateway,

WebSocketServer,

SubscribeMessage,

MessageBody,

OnGatewayConnection,

OnGatewayDisconnect,

} from '@nestjs/websockets';

import { Server, Socket } from 'socket.io';

import { DASHBOARD\_EVENTS } from './events/dashboard.events';

/\*\*

\* Gateway for handling real-time dashboard updates.

\*/

@WebSocketGateway({ namespace: '/dashboard', cors: true })

export class DashboardGateway implements OnGatewayConnection, OnGatewayDisconnect {

@WebSocketServer()

server: Server;

/\*\*

\* Handles a new client connection.

\* @param client - The connected socket client.

\*/

handleConnection(client: Socket) {

console.log(`Dashboard client connected: ${client.id}`);

}

/\*\*

\* Handles client disconnection.

\* @param client - The disconnected socket client.

\*/

handleDisconnect(client: Socket) {

console.log(`Dashboard client disconnected: ${client.id}`);

}

/\*\*

\* Listens for 'dashboard\_update' events and broadcasts the update to all clients.

\* @param payload - The data payload for the update.

\*/

@SubscribeMessage(DASHBOARD\_EVENTS.UPDATE)

handleDashboardUpdate(@MessageBody() payload: any) {

this.server.emit(DASHBOARD\_EVENTS.UPDATE, payload);

}

}

``



### 3. \*\*Notification Gateway (`notification.gateway.ts`)\*

This gateway manages real-time user notifications, allowing the server to push notifications to cliets.



```typescript

// src/websockets/notification.gateway.ts

import {

WebSocketGateway,

WebSocketServer,

SubscribeMessage,

MessageBody,

ConnectedSocket,

OnGatewayConnection,

OnGatewayDisconnect,

} from '@nestjs/websockets';

import { Server, Socket } from 'socket.io';

import { NOTIFICATION\_EVENTS } from './events/notification.events';

/\*\*

\* Gateway for handling real-time user notifications.

\*/

@WebSocketGateway({ namespace: '/notifications', cors: true })

export class NotificationGateway implements OnGatewayConnection, OnGatewayDisconnect {

@WebSocketServer()

server: Server;

/\*\*

\* Handles a new client connection.

\* @param client - The connected socket client.

\*/

handleConnection(client: Socket) {

console.log(`Notification client connected: ${client.id}`);

}

/\*\*

\* Handles client disconnection.

\* @param client - The disconnected socket client.

\*/

handleDisconnect(client: Socket) {

console.log(`Notification client disconnected: ${client.id}`);

}

/\*\*

\* Listens for 'notification\_new' events and emits the notification to the specific client.

\* @param payload - The notification data.

\* @param client - The socket client to send the notification to.

\*/

@SubscribeMessage(NOTIFICATION\_EVENTS.NEW)

handleNewNotification(@MessageBody() payload: any, @ConnectedSocket() client: Socket) {

client.emit(NOTIFICATION\_EVENTS.NEW, payload);

}

}

``



### 4. \*\*WebSocket Module (`websocket.module.ts`)\*

This module aggregates all WebSocket gateways, making them available throughout the applicaton.



```typescript

// src/websockets/websocket.module.ts

import { Module } from '@nestjs/common';

import { DashboardGateway } from './dashboard.gateway';

import { NotificationGateway } from './notification.gateway';

/\*\*

\* Module that encapsulates all WebSocket gateways.

\*/

@Module({

providers: [DashboardGateway, NotificationGateway],

exports: [DashboardGateway, NotificationGateway],

})

export class WebsocketModule {}

``



---

## 🔗 Integration into Other Modules or Servies

To utilize the WebSocket gateways in other parts of your application, you can inject them into services or controllers as neded.

\*\*Example: Emitting a Dashboard Update from a Servic\*\*



```typescript

// src/modules/analytics/analytics.service.ts

import { Injectable } from '@nestjs/common';

import { DashboardGateway } from '../../websockets/dashboard.gateway';

import { DASHBOARD\_EVENTS } from '../../websockets/events/dashboard.events';

/\*\*

\* Service for analytics-related operations.

\*/

@Injectable()

export class AnalyticsService {

constructor(private readonly dashboardGateway: DashboardGateway) {}

/\*\*

\* Processes new analytics data and emits a dashboard update.

\* @param data - The new analytics data.

\*/

processNewData(data: any) {

// Process the data...

this

**🔒 Security and Authentication**

Securing WebSocket connections is crucial to prevent unauthorized access

* **Guards**:NestJS allows the use of guards with WebSocket gateways, similar to HTTP routesHowever, instead of throwing HttpException, you should use WsException for WebSocket contexts citeturn0search1



@UseGuards(AuthGuard)

@SubscribeMessage('events')

handleEvent(client: Client, data: unknown): WsResponse<unknown> {

const event = 'events';

return { event, data };

}

```

- \*\*Middleware\*\*:For more granular control, especially during the handshake phase, consider implementing middleware to authenticate and authorize connections before they are established

---

## 🧪 Testing WebSocket Gateways

Testing ensures the reliability of your WebSocket gateway.

- \*\*Unit Testing\*\* Use NestJS's testing utilities to create unit tests for your gateway. Mock the `Socket` and `Server` instances to simulate client interaction.

- \*\*Integration Testing\*\* Set up integration tests to verify the end-to-end functionality of your WebSocket communication. Tools like `socket.io-client` can be used to simulate real client connection.

---

## 📦 Scalability Consideration

As your application grows, consider the following to maintain scalabiliy:

- \*\*Namespace Segmentation\*: Use namespaces to separate concerns and reduce unnecessary event broadcastig.

- \*\*Load Balancing\*: Implement load balancing strategies to distribute WebSocket connections across multiple server instancs.

- \*\*Redis Adapter\*: For horizontal scaling, use a Redis adapter to synchronize events across different server instancs.

---

## 🧩 Integration with Other Moduls

To utilize WebSocket gateways in other modules or services, inject them as providrs:



```typescript

// src/modules/analytics/analytics.service.ts

import { Injectable } from '@nestjs/common';

import { DashboardGateway } from '../../websockets/dashboard.gateway';

import { DASHBOARD\_EVENTS } from '../../websockets/events/dashboard.events';

@Injectable()

export class AnalyticsService {

constructor(private readonly dashboardGateway: DashboardGateway) {}

/\*\*

\* Processes new analytics data and emits a dashboard update.

\* @param data - The new analytics data.

\*/

processNewData(data: any) {

// Process the data...

this.dashboardGateway.server.emit(DASHBOARD\_EVENTS.UPDATE, data);

}

}

``



Ensure that `DashboardGateway` is exported from the `WebsocketModule` and that `WebsocketModule` is imported into the module where `AnalyticsService` is defied.

---

If you have any further questions or need assistance with another directory, feel free to ask!