

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бек-энд разработка

Отчет

Лабораторная работа 6

Выполнил:

Прокопец Семен

К3439

**Проверил:
Добряков Д. И.**

Санкт-Петербург

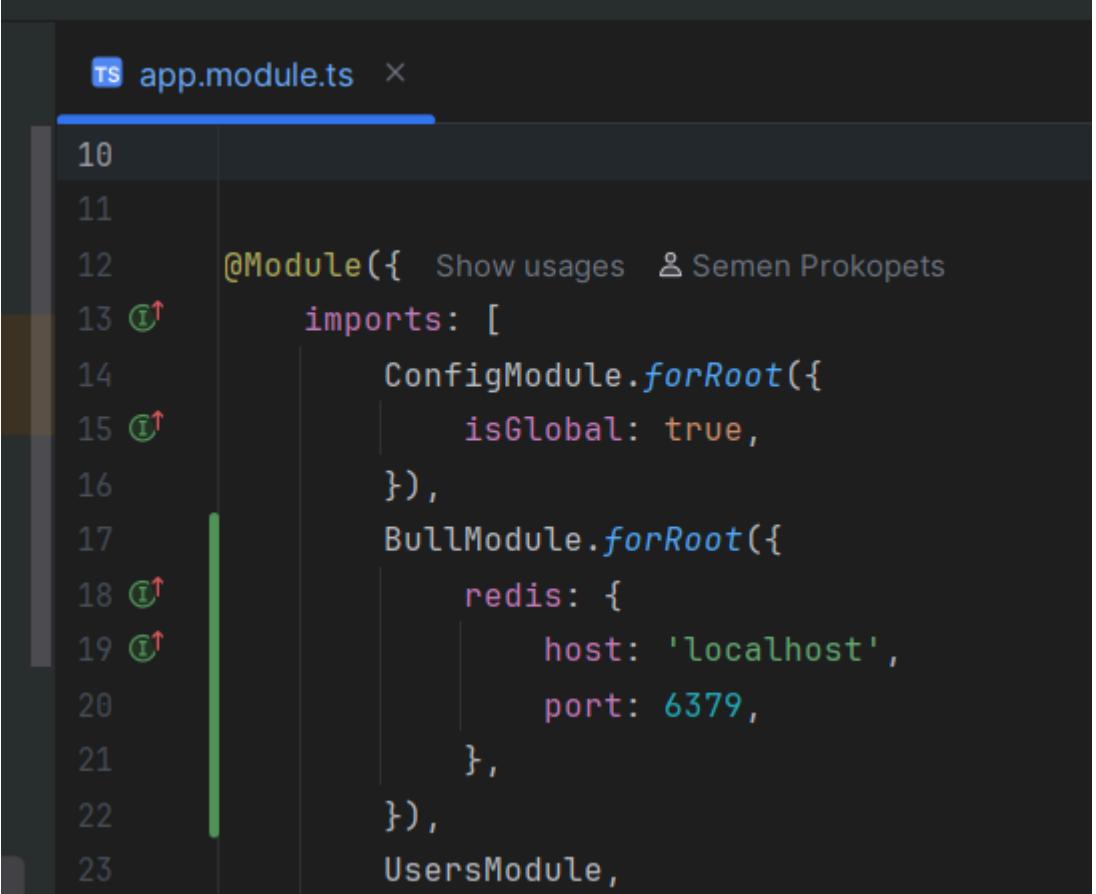
2025 г.

Задача

Добавить очередь

Ход работы

подключаем Bull в [app.module.ts](#)



The screenshot shows a code editor window for a file named 'app.module.ts'. The code is written in TypeScript and defines an @Module block. The imports section includes ConfigModule.forRoot() and BullModule.forRoot(). The BullModule.forRoot() configuration specifies a redis connection with host 'localhost' and port 6379. Other imports listed are UsersModule and another module that is partially visible. The code is numbered from 10 to 23.

```
10
11
12     @Module({
13         imports: [
14             ConfigModule.forRoot({
15                 isGlobal: true,
16             }),
17             BullModule.forRoot({
18                 redis: {
19                     host: 'localhost',
20                     port: 6379,
21                 },
22             }),
23             UsersModule,
```

создаем

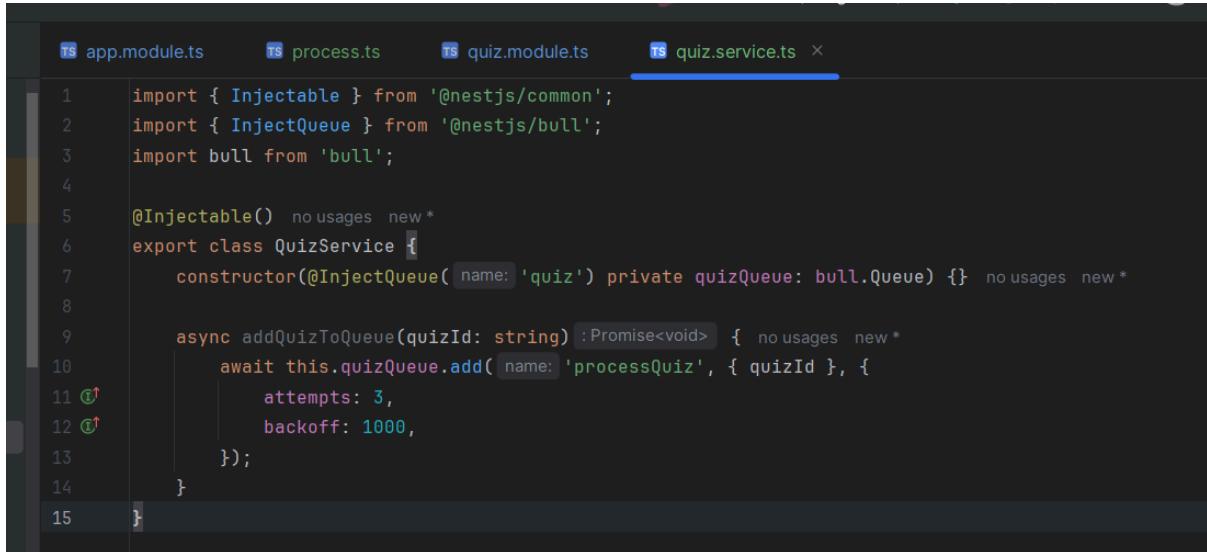
очередь

```
ts app.module.ts      ts process.ts ×
1 import { Process, Processor } from '@nestjs/bull';
2 import bull from 'bull';
3
4 @Processor({ queueName: 'quiz' }) no usages new *
5 export class QuizProcessor {
6     @Process({ name: 'processQuiz' }) no usages new *
7     async handleProcessQuiz(job: bull.Job<{ quizId: string }>): Promise<{ status: string; quizId: string ... }> {
8         console.log(`Обработка квиза: ${job.data.quizId}`);
9
10        await new Promise(resolve: (value: unknown) => void) => setTimeout(resolve, delay: 3000);
11
12        return { status: 'completed', quizId: job.data.quizId };
13    }
14}
```

регистрируем очередь в модуле

```
ts app.module.ts      ts process.ts      ts quiz.module.ts ×
1 > import ...
2
3
4
5
6
7
8     @Module({
9         imports: [
10             BullModule.registerQueue({
11                 name: 'quiz',
12             }),
13         ],
14     })
15 }
```

добавляем логику в сервис



```
1 import { Injectable } from '@nestjs/common';
2 import { InjectQueue } from '@nestjs/bull';
3 import bull from 'bull';
4
5 @Injectable() no usages new*
6 export class QuizService {
7     constructor(@InjectQueue('quiz') private quizQueue: bull.Queue) {} no usages new*
8
9     async addQuizToQueue(quizId: string): Promise<void> { no usages new*
10         await this.quizQueue.add('processQuiz', { quizId }, {
11             attempts: 3,
12             backoff: 1000,
13         });
14     }
15 }
```

и в контроллер

```
@Post(':id/process')
async processQuiz(@Param('id') id: string) {
    await this.quizService.addQuizToQueue(id);
    return { message: 'Задача добавлена в очередь' };
}
```

Вывод: Научились создавать и работать с очередью Bull