

## **Разработка бэкенда веб-сервиса для добавления записей с геопривязкой и фотоматериалами на основе NestJS**

*Андриенко Иван Сергеевич*

*Приамурский государственный университет имени Шолом-Алейхема*  
*Студент*

*Пащенко Екатерина Александровна*

*Приамурский государственный университет имени Шолом-Алейхема*  
*Студент*

### **Аннотация**

В статье рассматривается реализация серверной части веб-сервиса для добавления и хранения записей с географической привязкой и фотоматериалами. Описываются механизмы аутентификации администратора на основе JWT, защищённая обработка multipart-запросов, загрузка и нормализация файлов изображений, а также сохранение данных в базе с использованием пространственных типов. Показано, что выбранная архитектура обеспечивает безопасность, надёжность хранения данных и возможность дальнейшего масштабирования сервиса.

**Ключевые слова:** backend-разработка, NestJS, JWT-аутентификация, загрузка файлов, REST API, PostGIS, геоданные, веб-сервис.

## **Development of a web service backend for adding geolocation records and photographic materials based on NestJS**

*Andrienko Ivan Sergeevich*

*Sholom-Aleichem Priamursky State University*  
*Student*

*Paschenko Ekaterina Alexandrovna*

*Sholom-Aleichem Priamursky State University*  
*Student*

### **Abstract**

The article discusses the implementation of the server side of a web service for adding and storing geo-referenced records and photographic materials. JWT-based administrator authentication mechanisms, secure multipart request processing, image file loading and normalization, and data storage in a database using spatial types are described. It is shown that the chosen architecture ensures the security, reliability of data storage and the possibility of further scaling of the service.

**Keywords:** backend development, NestJS, JWT authentication, file upload, REST API, PostGIS, geodata, web service.

## **1 Введение**

### **1.1 Актуальность**

Актуальность исследования обусловлена необходимостью создания надёжных и защищённых серверных решений для веб-сервисов, работающих с чувствительными данными и пользовательскими загрузками. В задачах, связанных с хранением пространственных данных и фотоматериалов, особое значение приобретают вопросы аутентификации, валидации входных данных и безопасной обработки файлов. Использование современных backend-фреймворков и стандартных механизмов безопасности позволяет повысить устойчивость системы и снизить риски несанкционированного доступа.

### **1.2 Обзор исследований**

Интеграция картографических сервисов в программные приложения на основе React и TypeScript находит отражение в современной научной литературе. В работе А.А. Севера рассматривается использование API карт Google Maps, Yandex.Maps и 2GIS в приложении на JavaFX, где анализируются особенности подключения различных картографических сервисов, их программные интерфейсы и практические аспекты реализации отображения карт и маркеров в десктопной среде [1]. Н.И. Филиппов и А.В. Лошаков выполняют сравнительный анализ сервисов Google Maps и Yandex.Maps, оценивая их функциональные возможности, точность картографических данных, удобство интеграции и особенности использования в прикладных задачах [2]. В статье Д.А. Хайруллина и М.Р. Богданова рассматриваются особенности применения статической типизации в React-приложениях, включая методики типизации компонентов, хуков и состояния, а также влияние строгой типизации на надёжность пользовательских интерфейсов [3]. К.Р. Удальцов исследует вопросы обеспечения качества кода в командной разработке, предлагая формализацию шаблонов безопасных pull request-ов для проектов на React и TypeScript в рамках Git Flow [4]. Особое внимание созданию библиотек компонентов на основе TypeScript уделяется в работе Т.М. Воронко, где описываются принципы модульности, повторного использования кода и проектирования архитектуры компонентов для масштабируемых React-решений [5].

### **1.3 Цель исследования**

Цель исследования – разработать и описать серверную часть веб-сервиса для добавления записей с географическими координатами и фотоматериалами, обеспечивающую защищённую авторизацию, корректную обработку файлов и сохранение данных в пространственной базе данных.

## **2 Материалы и методы**

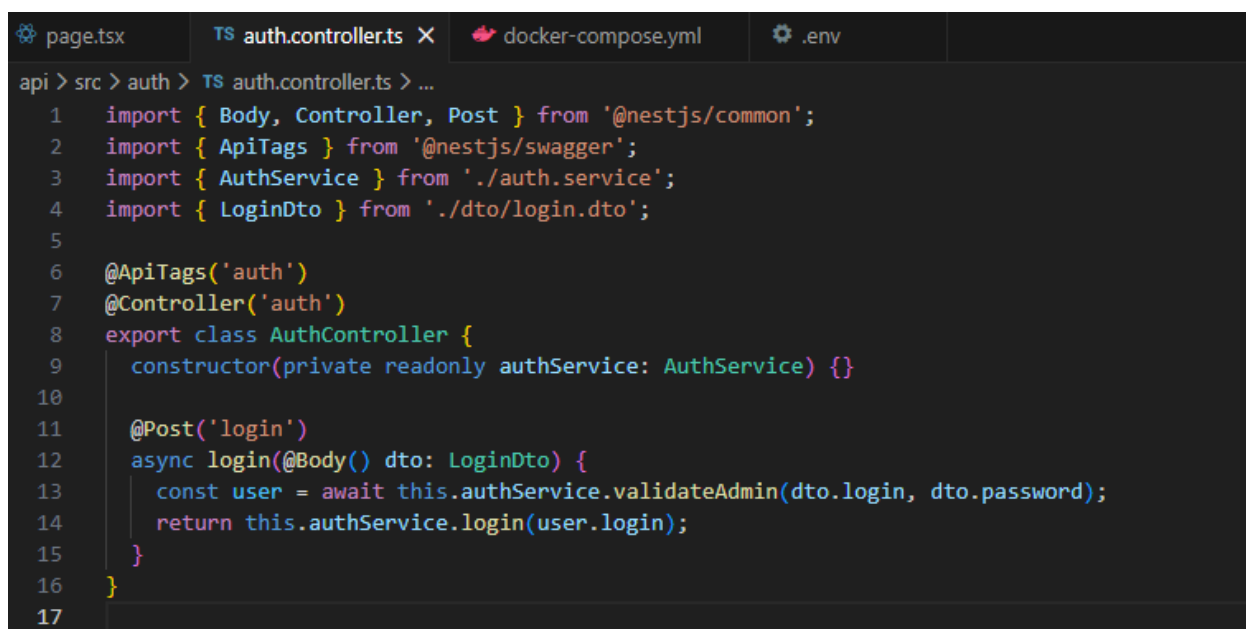
В качестве серверной платформы использовался фреймворк NestJS. Для аутентификации применялся механизм JWT с проверкой ролей. Загрузка файлов реализована с использованием middleware multer и хранения данных на файловой системе сервера. Для валидации входных данных применялись

DTO и декораторы class-validator. Хранение записей осуществлялось в реляционной базе данных PostgreSQL с расширением PostGIS для работы с географическими координатами.

### 3 Результаты и обсуждения

Для защиты операций добавления записей в систему реализован механизм аутентификации администратора на основе JSON Web Token. Аутентификация выполняется через отдельный эндпоинт, принимающий логин и пароль, после чего происходит их проверка с использованием значений из переменных окружения или хешированного пароля.

В случае успешной валидации формируется JWT-токен, содержащий информацию о роли пользователя. Токен подписывается секретным ключом и используется для последующей авторизации защищённых эндпоинтов. Такой подход позволяет изолировать операции добавления данных от публичного доступа и обеспечить базовый уровень безопасности сервиса (рис. 1).



```
api > src > auth > TS auth.controller.ts > ...
1  import { Body, Controller, Post } from '@nestjs/common';
2  import { ApiTags } from '@nestjs/swagger';
3  import { AuthService } from './auth.service';
4  import { LoginDto } from './dto/login.dto';
5
6  @ApiTags('auth')
7  @Controller('auth')
8  export class AuthController {
9    constructor(private readonly authService: AuthService) {}
10
11    @Post('login')
12    async login(@Body() dto: LoginDto) {
13      const user = await this.authService.validateAdmin(dto.login, dto.password);
14      return this.authService.login(user.login);
15    }
16  }
17
```

Рис. 1. Логика аутентификации администратора и формирования JWT-токена

Для проверки подлинности запросов используется JWT-стратегия, извлекающая токен из HTTP-заголовка Authorization. На этапе обработки запроса выполняется проверка подписи токена и роли пользователя. Доступ к эндпоинтам добавления записей разрешается только при наличии валидного токена с ролью администратора. Такой механизм предотвращает несанкционированное создание записей и исключает возможность подмены данных со стороны неавторизованных пользователей (рис. 2).

```
api > src > auth > TS jwt.strategy.ts > ...
1 import { Injectable } from '@nestjs/common';
2 import { ConfigService } from '@nestjs/config';
3 import { PassportStrategy } from '@nestjs/passport';
4 import { ExtractJwt, Strategy } from 'passport-jwt';
5
6 type JwtPayload = {
7   sub: string;
8   role: string;
9   login: string;
10 };
11
12 @Injectable()
13 export class JwtStrategy extends PassportStrategy(Strategy) {
14   constructor(configService: ConfigService) {
15     super({
16       jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
17       ignoreExpiration: false,
18       secretOrKey: configService.get<string>('JWT_SECRET', 'dev-secret'),
19     });
20   }
21
22   async validate(payload: JwtPayload) {
23     if (payload.role !== 'admin') return null;
24     return { userId: payload.sub, login: payload.login, role: payload.role };
25   }
26 }
27
```

Рис. 2. JWT-стратегия и проверка роли администратора

Добавление записи реализовано в виде защищённого REST-эндпоинта, принимающего multipart-запрос с текстовыми данными и фотографиями. Эндпоинт доступен только при наличии валидного JWT-токена. В процессе обработки запроса выполняется извлечение координат, нормализация входных параметров и формирование ссылок на загруженные файлы. Реализация предусматривает строгую проверку наличия обязательных координат, что исключает сохранение некорректных географических данных (рис. 3).

```

api > src > graves > TS graves.controller.ts > ...
22 import { Grave } from './entities/grave.entity';
23 import { GravesService } from './graves.service';
24 import { AuthGuard } from '@nestjs/passport';
25
26 const decodeOriginalName = (original: string) => {
27   try {
28     // Multer декодирует имя в latin1; возвращаем в utf8
29     return Buffer.from(original, 'latin1').toString('utf8');
30   } catch {
31     return original;
32   }
33 };
34
35 const makeSafeFilename = (original: string) => {
36   const decoded = decodeOriginalName(original);
37   const ext = path.extname(decoded) || '';
38   const base = path.basename(decoded, ext).replace(/\/s+/g, '_');
39   const sanitized = base.replace(/[^a-zA-Z0-9_\.\-яА-ЯЕ_]/g, '') || 'file';
40   return `${sanitized}_${Date.now()}${ext}`;
41 };
42
43 @ApiTags('graves')
44 @Controller('graves')
45 export class GravesController {
46   constructor(
47     private readonly gravesService: GravesService,
48     private readonly configService: ConfigService,
49   ) {}
50
51   @ApiBearerAuth()
52   @UseGuards(AuthGuard('jwt'))
53   @Post()
54   create(@Body() dto: CreateGraveDto): Promise<Grave> {
55     return this.gravesService.create(dto);
56   }
57
58   @Get()
59   findAll(@Query() query: ListGravesQueryDto): Promise<Grave[]> {
60     return this.gravesService.findAll({
61       bbox: this.parseBbox(query.bbox),

```

Рис. 3. Защищённый эндпоинт добавления записи с файлами

После обработки входных данных формируется объект записи, содержащий текстовую информацию, массив URL фотографий и географическую точку. Координаты преобразуются в формат Point, совместимый с расширением PostGIS.

Использование пространственного типа данных позволяет в дальнейшем выполнять географические запросы и эффективно работать с координатами в контексте картографического сервиса (рис. 4).

```

10 type Bbox = [number, number, number, number]; // [minLng, minLat, maxLng, maxLat]
11
12 @Injectable()
13 export class GravesService {
14   constructor() {
15     @InjectRepository(Grave)
16     private readonly gravesRepo: Repository<Grave>,
17     private readonly configService: ConfigService,
18   } {}
19
20   async create(dto: CreateGraveDto): Promise<Grave> {
21     const grave = this.gravesRepo.create({
22       name: dto.name,
23       bio: dto.bio,
24       tags: dto.tags ?? [],
25       photoUrls: dto.photoUrls ?? [],
26       location: {
27         type: 'Point',
28         coordinates: [dto.location.lon, dto.location.lat],
29       },
30     });
31     return this.gravesRepo.save(grave);
32   }
33
34   async findAll(options: {
35     bbox?: Bbox;
36     q?: string;
37     limit?: number;
38     page?: number;
39     hasPhoto?: boolean;
40   }): Promise<Grave[]> {
41     const qb = this.gravesRepo
42       .createQueryBuilder('grave')
43       .orderBy('grave.updatedAt', 'DESC');

```

Рис. 4. Создание маркеров и обработка клика

Проведённая реализация показала, что использование JWT-аутентификации в сочетании с защищёнными эндпоинтами и строгой валидацией данных позволяет создать надёжную серверную часть для добавления записей с географической привязкой. Обработка файлов и нормализация имён обеспечивают устойчивую работу сервиса при загрузке изображений, а применение PostGIS позволяет корректно хранить и использовать пространственные данные (рис. 5).

Админ-панель

### Управление записями

Список Добавить Выйти

Поиск по имени или описанию ☐ Есть фото Обновить

| Имя                 | Координаты            | Фото | Обновлено            | Действия              |
|---------------------|-----------------------|------|----------------------|-----------------------|
| Алексей Иванов #1   | 48.765736, 132.949169 | 1    | 16.12.2025, 16:15:13 | Редактировать Удалить |
| Мария Петрова #2    | 48.765895, 132.953700 | 1    | 16.12.2025, 16:15:13 | Редактировать Удалить |
| Исаак Левин #3      | 48.766273, 132.955430 | 1    | 16.12.2025, 16:15:13 | Редактировать Удалить |
| Софья Гольдман #4   | 48.765999, 132.952375 | 1    | 16.12.2025, 16:15:13 | Редактировать Удалить |
| Яков Берман #5      | 48.765726, 132.950091 | 1    | 16.12.2025, 16:15:13 | Редактировать Удалить |
| Давид Кац #6        | 48.764382, 132.956659 | 1    | 16.12.2025, 16:15:13 | Редактировать Удалить |
| Елизавета Левина #7 | 48.764801, 132.951129 | 1    | 16.12.2025, 16:15:13 | Редактировать Удалить |
| Семён Виноградов #8 | 48.766419, 132.952162 | 1    | 16.12.2025, 16:15:13 | Редактировать Удалить |
| Анна Шапиро #9      | 48.766880, 132.953014 | 1    | 16.12.2025, 16:15:13 | Редактировать Удалить |

Рис. 5. Админ панель веб-сервиса

#### **4 Выводы**

В ходе работы была реализована серверная часть веб-сервиса, обеспечивающая защищённое добавление записей с координатами и фотоматериалами. Разработаны механизмы аутентификации администратора на основе JWT, защищённые REST-эндпоинты, обработка и хранение файлов изображений, а также сохранение данных в пространственной базе данных.

Использование современных backend-технологий и стандартных механизмов безопасности позволило создать устойчивое и расширяемое решение, пригодное для использования в веб-картографических и информационных системах, работающих с географическими данными и пользовательскими загрузками.

#### **Библиографический список**

1. Север А.А. Использование API карт Google Maps, Yandex.Maps и 2GIS в приложении JavaFX // Научные труды магистрантов и аспирантов. Нижневартовск, 2019. С. 265–268.
2. Филиппов Н.И., Лошаков А.В. Сравнительный анализ картографических сервисов Google Maps и Yandex Maps // Актуальные научные исследования в современном мире. 2021. № 4-3 (72). С. 128–133.
3. Хайруллин Д.А., Богданов М.Р. Изучение использования TypeScript в React-приложениях, преимущества статической типизации, подходы к типизации компонентов, хуков и состояния // Международный журнал гуманитарных и естественных наук. 2024. № 12-3 (99). С. 233–238.
4. Удальцов К.Р. Формализация шаблонов безопасных pull request-ов для React-приложений с TypeScript в командах с Git Flow // Актуальные проблемы социально-гуманитарного и научно-технического знания. 2025. № 4 (44). С. 35–36.
5. Воронко Т.М. Создание React-библиотеки с использованием TypeScript // Электронные системы и технологии. Материалы 58-й научной конференции аспирантов, магистрантов и студентов БГУИР. Минск, 2022. С. 200–202.