Утверждаю

Руководитель организации

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, печать организации)

« \_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Учреждение образования

«Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 05 01-03 «Информационные системы и технологии»

ОТЧЕТ

по производственной технологической практике

на \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование организации, сроки практики)

Исполнитель

студент курса \_\_ группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (Ф.И.О.)

Руководитель практики

от организации

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, печать предприятия) (подпись, дата) (Ф.И.О.)

Руководитель практики

от кафедры

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, уч. звание) (подпись, дата). (Ф.И.О.)

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Минск 2024 г

**Содержание**

[Введение 4](#_Toc171979534)

[1. Общая характеристика структуры предприятия прохождения практики. 5](#_Toc171979535)

[2. Общая характеристика проекта 6](#_Toc171979536)

[3. Характеристика и описание информационных инструментов, использованных в течение практики. 7](#_Toc171979537)

[4. Характеристика и описание разделов проекта 9](#_Toc171979538)

[4.1 Конфигурация 9](#_Toc171979539)

[4.2 Реализация REST API. Контроллеры 10](#_Toc171979540)

[4.3 Сервисы 11](#_Toc171979541)

[4.4 Взаимодействие с БД 12](#_Toc171979542)

[Заключение 13](#_Toc171979543)

[Список использованных источников информации 14](#_Toc171979544)

# **Введение**

Технологическая практика является неотъемлемой частью образовательной программы, предназначенной для подготовки дипломированных специалистов. Её значимость заключается в формировании комплекса знаний и навыков у будущих специалистов. Целью данной практики является закрепление теоретических знаний и приобретение опыта в решении реальных задач.

В ходе технологической практики студенты должны освоить:

* организацию и управление деятельностью подразделения;
* планирование этапов работы над проектом;
* действующие стандарты, документацию и инструкции по разработке проекта;
* хорошие практики от опытных специалистов.

Кроме того, студентам предстоит овладеть особенностями использования выбранной платформы для разработки веб-приложений.

Основные задачи практики:

* ознакомление с организационной структурой компании ООО «Фабрика инноваций и решений».
* анализ информационных ресурсов и информационных технологий в компании.
* работа в команде и навыки распределения задач с определением приоритетов.
* выполнение поставленных задач.
* составление отчёта в соответствии с программой практики на основе проделанной работы и собранной информации.

Технологическая практика будет проходить в компании ООО «Фабрика инноваций и решений» в течение периода с 17 июня 2024 года по 12 июля 2024 года.

Общая характеристика структуры предприятия прохождения практики.

В структуре компании можно выделить:

* исполнительный директор (CEO): руководитель компании, принимающий стратегические решения и ответственный за общее управление компанией;
* отделы или подразделения:
* разработка ПО (Software Development): команда разработчиков, которая занимается созданием программного обеспечения и веб-приложений;
* тестирование (Quality Assurance): специалисты по тестированию и контролю качества программного обеспечения;
* продажи и маркетинг (Sales & Marketing): команда, отвечающая за поиск клиентов, продвижение продуктов и услуг компании;
* управление проектами (Project Management): отдел, занимающийся планированием, контролем и управлением проектами компании;
* ИТ-поддержка (IT Support): специалисты, обеспечивающие поддержку и обслуживание информационной инфраструктуры компании;
* аналитика (Business Analysis): команда аналитиков, занимающаяся анализом бизнес-процессов и потребностей клиентов;
* HR и административный отдел (Human Resources & Administration): отдел, ответственный за управление персоналом, подбор кадров и административные вопросы.
* команда разработчиков: может включать различные специализации, такие как фронтенд, бэкенд, мобильная разработка и другие;
* клиенты и партнеры: список клиентов и партнеров, с которыми компания сотрудничает, для выполнения проектов и предоставления услуг.

Также имеется в виду то, что для выполнения какой-либо задачи формируется команда из нужных для этой задачи людей.

Работа ведётся в основном с зарубежными заказчиками, поэтому основным языком компании является английский, но так же компания начинает использовать немецкий язык.

Общая характеристика проекта

Информационный проект, выполненный в рамках технологической практики, представляет собой RESTful сервис для публикации постов. Основные этапы реализации проекта включали:

Проект включал в себя следующие основные задачи:

* Проектирование структуры базы данных;
* Выбор системы управления базами данных, наиболее подходящей для задач управления проектами;
* Разработка скриптов для создания и инициализации базы данных;
* Реализация механизмов миграции и управления версиями базы данных;
* Запуск и тестирование приложения в контейнерах Docker;
* Создание веб-платформы, позволяющей создавать, отслеживать и управлять постами, подписками лайками и комментариями, включая функции фильтрации и сортировки по различным параметрам.

В проекте были реализованы две роли с различными уровнями доступа:

* Администратор (особый уровень привилегий над пользовательскими возможностями, учитывая доступ к несобственным ресурсам);
* Пользователь (создание, обновление и удаление постов, лайки, подписки, комментарии).

Система авторизации и аутентификации была основана на JWT-токенах, которые хранились в хранилище Redis для повышения безопасности. RESTful сервис также оборудован документацией Swagger и содержит Unit и E2E тесты для каждого метода в каждом контроллере и отдельных популярных циклов задач, обеспечивая высокое качество кода. Для контроля качества кода, соблюдения стандартов разработки и отслеживания прогресса использовались такие инструменты, как Git, GitHub, ClickUp и Clockify.

Характеристика и описание информационных инструментов, использованных в течение практики.

Для разработки проекта была выбрана платформа Nest JS, основанная на Node JS и TypeScript, работающая внутри на фреймворке Fastify, обеспечивающего огромный прирост производительности в обработке запросов, вместо привычного Express.

Nest JS — это модульная платформа для разработки программного обеспечения с открытым исходным кодом, предназначенная для создания масштабируемых и эффективных серверных приложений. Основана на TypeScript и работает на платформе Node JS. Nest JS обладает рядом ключевых особенностей:

* кроссплатформенность: Nest JS совместима с различными операционными системами, такими как Windows, macOS и Linux, и может выполняться в среде Node JS;
* открытый исходный код: Nest JS разрабатывается на платформе GitHub и доступен по лицензии MIT, что позволяет разработчикам свободно использовать, изменять и распространять его;
* современные парадигмы: Nest JS поддерживает современные методы разработки, такие как асинхронное программирование, использование декораторов и инъекции зависимостей;
* производительность: благодаря оптимизированным архитектурным решениям, Nest JS обеспечивает высокую производительность и эффективность работы приложений;
* поддержка различных сред: ваш код будет работать на различных операционных системах и архитектурах без необходимости внесения дополнительных изменений;
* инструменты командной строки: Nest JS предоставляет удобные инструменты командной строки для локальной разработки и интеграции в процесс непрерывной интеграции.
* гибкое развертывание: Nest JS может быть встроена в ваше приложение или развернута параллельно с другими системами. Также можно использовать с контейнерами Docker для удобного развертывания.

Выбор платформы Nest JS обеспечивает разработчикам удобную и мощную среду для создания современных и высокоэффективных серверных приложений на платформе Node JS.

В проекте также использовалась платформа ClickUp. Ниже представлено описание этой платформы.

ClickUp – это универсальный инструмент для управления проектами, который объединяет управление задачами, документацией, целями, календарями и даже электронной почтой в одном приложении. Использование ClickUp позволяет значительно повысить эффективность организации и управления работой команды.

Универсальность: ClickUp подходит для команд любого размера и специализации, предоставляя инструменты для гибкой настройки рабочих процессов.

Разработка: ClickUp был разработан компанией Mango Technologies, базирующейся в Сан-Диего, Калифорния.

Платформонезависимость: ClickUp доступен для различных операционных систем, включая Windows, macOS, Linux, а также в виде мобильных приложений для iOS и Android.

Многоязычность: Платформа поддерживает несколько языков, включая английский, немецкий, испанский, французский и другие.

Интеграция: ClickUp может быть интегрирован с множеством других инструментов, таких как Google Calendar, Slack, GitHub, GitLab и другие.

Поддержка различных баз данных: Встроенные инструменты ClickUp позволяют взаимодействовать с различными типами данных и интеграциями.

Эти особенности делают ClickUp мощным решением для управления проектами и задачами в различных условиях и с разнообразными требованиями.

Отслеживание времени выполнения: Clockify обеспечивает удобный интерфейс и поддерживает различные роли, за счёт чего и организовывалось отслеживание успехов целой команды людей.

Во время разработки, когда кто-то заканчивает работу над своим заданием, он делает pull-request, который попадает на просмотр Team lead. Team lead в свою очередь делает сode review и помогает решить некоторые спорные ситуации. И только после просодобрения Tam lead’ом эта часть попадает в основную ветку.

Для разработки backend была использована интегрированная среда разработки (IDE) Visual Studio Code от Microsoft.

Характеристика и описание разделов проекта

Для разработки проекта использовались фреймворки Nest JS и Fastify; язык, который используется в основе данного фреймворка – Typescript.

## **4.1 Конфигурация**

Для работы приложения необходимы секретные данные, такие как строка подключения к базе данных или секрет для JWT-токена, которые хранятся в .env файлах. Пример содержимого .env файла представлен на листинге 4.1.

PORT=5000

POSTGRES\_HOST=postgres\_cp

POSTGRES\_USER=postgres

POSTGRES\_DB=Test

POSTGRES\_PASSWORD= admin

POSTGRES\_PORT=5432

PRIVATE\_KEY=secret\_key

REDIS\_HOST=redis\_cp

REDIS\_PORT=6379

DATABASE\_URL="postgresql://postgres:whoami@postgreS\_cp:5432/InstInnow?schema=public"

Листинг 4.1 – Пример файла .env

В файле tsconfig.json определены конфигурационные настройки для Typescript, а в файле nest-cli json находятся настройки для nest-cli.

Файлы eslint и prettier задают настройки для контроля качества кода, которые должны быть соблюдены при разработке программного продукта.

Метод в классе main.ts отвечает за настройку сервера, включая установку глобальных middleware и запуск сервера на порту, указанном в .env файле. Ниже представлен листинг 4.2 с кодом файла main.ts.

import { NestFactory } from '@nestjs/core';

import { AppModule } from './app.module';

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

import \* as cookieParser from 'cookie-parser';

async function bootstrap() {

  const PORT = process.env.PORT || 5000;

  const app = await NestFactory.create(AppModule);

  app.useGlobalPipes(new ValidationPipe());

  app.setGlobalPrefix('api');

  app.use(cookieParser());

  await app.listen(PORT, () => console.log(`Launched on ${PORT}`));}

bootstrap();

Листинг 4.2 – Файл main.ts

Также в этом методе устанавливается глобальный префикс.

## **4.2 Реализация REST API. Контроллеры**

Контроллеры в Nest JS представляют собой API. Каждый контроллер имеет шаблон, который описывает его структуру и характеристики. Для этого над классом контроллера указывается декоратор, а затем внутри класса идут методы, каждый из которых помечен другим декоратором, показывающим, какой тип HTTP-метода соответствует методу класса, какой тип данных он принимает и какой возвращает. На листинге 4.3 приведен пример контроллера для регистрации и авторизации.

@Controller('auth')

export *class* AuthController {

*constructor*(*private* *authService*: AuthService) {}

  @Post('/login')

  login(@Body() *loginUserDto*: LoginUserDto) {

    return *this*.authService.login(loginUserDto);

  }

  @Post('/registration')

  registration(@Body() *createUserDto*: CreateUserDto) {

    return *this*.authService.registration(createUserDto);

  }

  @Post('/refresh-token')

  getRefreshedTokens(@Body() *refreshTokenDto*: RefreshTokenDto) {

    return *this*.authService.getRefreshedTokens(refreshTokenDto);

  }

  @Get('/logout')

  logout(@Req() *req*: *any*) {

*const* id = req.user.id;

    return *this*.authService.deleteRefreshTokens(id);

  }

}

Листинг 4.3 – Контроллер для авторизации и регистрации

Nest JS автоматически валидирует входные данные, используя класс DTO, который содержит ограничения на входные данные, указанные с помощью декораторов. Если входные данные не соответствуют этим ограничениям, глобальный middleware ValidationPipe возвращает ответ со статусом ошибки 400 и пояснительным текстом. Ниже представлен листинг 4.4 с классом LoginUserDto.

import { IsNotEmpty } from 'class-validator';

export *class* LoginUserDto {

  @IsNotEmpty()

*readonly* login: *string*;

  @IsNotEmpty()

*readonly* password: *string*;

}

Листинг 4.4 – Класс LoginUserDto

Контроллеры также используют механизм Dependency Injection, чтобы встраивать сервисы, которые в них используются.

## **4.3 Сервисы**

Сервисы – классы, которым контроллеры делегируют выполнение различных действий, после поступления на них запроса. В Nest JS такие классы являются провайдерами (Providers) и помечены декоратором @Injectable. Пример сервиса для бронирования представлен на листинге 4.5.

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

import { CreateReservationDto } from './reservations/dto/create-reservation.dto';

import { UpdateReservationDto } from './reservations/dto/update-reservation.dto';

import { ReservationsRepository } from './reservations.repository';

@Injectable()

export class ReservationsService {

  constructor(private readonly reservationsRepository: ReservationsRepository) { }

  create(createReservationDto: CreateReservationDto, userId: string) {

    return this.reservationsRepository.create({

      ...createReservationDto,

      timestamp: new Date(),

      userId

    });

  }

  findAll() {

    return this.reservationsRepository.find({});

  }

  findOne(\_id: string) {

    return this.reservationsRepository.findOne({ \_id });

  }

  update(\_id: string, updateReservationDto: UpdateReservationDto) {

    return this.reservationsRepository.findOneAndUpdate({ \_id }, { $set: updateReservationDto });

  }

  remove(\_id: string) {

    return this.reservationsRepository.findOneAndDelete({ \_id });

  }

}

Листинг 4.5 – Сервис для тегов

Большинство сервисов лишь выполняют различные действия с репозиториями, иногда лишь делая небольшие преобразования в объектах, чтобы ORM-система смогла правильно их обработать. Работа сервисов совершается в неблокирующем асинхронном режиме. В конструктор сервиса так же встраиваются различные модели.

## **4.4 Взаимодействие с БД**

Для работы с базой данных в данном проекте применяется ORM Prisma. На первом этапе необходимо определить сущности, которые отражают структуру таблиц базы данных. После этого, используя функционал Prisma, можно проводить различные операции с данными (CRUD-операции) через специальные методы, предоставляемые клиентом Prisma.

При инициализации приложения Prisma автоматически создает объекты классов для каждой сущности, включая методы для создания, обновления и удаления данных. Каждый класс сущности включает в себя описание полей и связей.

Использование Prisma избавляет разработчика от необходимости написания SQL-кода, поскольку все запросы формируются автоматически на основе определенных сущностей. Кроме того, Prisma поддерживает управление транзакциями и миграциями, что обеспечивает гибкое управление изменениями в структуре данных базы данных.

Заключение

В результате проделанной работы во время производственной практики была изучена структура предприятия ООО «Фабрика инноваций и решений». Были изучены основные нормативно-правовые акты, углублены и закреплены знания, полученные при изучении специальных дисциплин.

В качестве задания был разработана платформа для задания вопросов, получения ответов. Было реализовано разделение ролей на User и Admin. C целью создания качественного решения, были проанализированы различные подходы и инструменты для разработки backend части. В качестве платформы backend был выбран Nest JS. Были улучшены практические навыки написания хорошего кода.

Было улучшено понятие многоуровневой архитектуры приложения и знания протокола http. Знания, полученные во время обучения в университете, были применены на практике.

Список использованных источников информации

Prisma [Электронный ресурс]. – Режим доступа: https://www.prisma.io – Дата доступа: 26.06.2024.

Что такое Node JS [Электронный ресурс]. – Режим доступа: https://metanit.com/web/nodejs/1.1.php – Дата доступа: 01.07.2024.

Документация по Nest JS [Электронный ресурс]. – Режим доступа: https://Nest JS.com/ –Дата доступа: 17.06.2024.

Асинхронный веб, или что такое веб-сокеты [Электронный ресурс]. – Режим доступа: https://tproger.ru/translations/what-are-web-sockets/ – Дата доступа: 02.07.2024.

Socket.IO [Электронный ресурс]. – Режим доступа: https://socket.io/ – Дата доступа: 21.06.2024.

N-layer архитектура [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier – Дата доступа: 04.07.2024.