Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей  
  
Кафедра программного обеспечения информационных технологий  
  
Дисциплина: Современные платформы программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к курсовой работе  
на тему

«To-do list»

БГУИР КП 1-40 01 01 23 ПЗ

Студент: гр 051004 Сасим Е.С.

Руководитель: Вашкевич М.А.

Минск 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc133634938)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc133634939)

[1.1 Обзор аналогов 4](#_Toc133634940)

[1.2 Формирование требований к проектируемому ПС 4](#_Toc133634941)

[2 ПРОЕКТИРОВНИЕ ПРОГРАММНОГО СРЕДСТВА 6](#_Toc133634942)

[2.1 Структура приложения. 6](#_Toc133634943)

[2.2 Взаимодействие клиентской и серверной частей 7](#_Toc133634944)

[2.3 Проектирование базы данных 9](#_Toc133634945)

[3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА 11](#_Toc133634946)

[3.1 Выбор среды разработки 11](#_Toc133634947)

[3.2 Выбор языка программирования и платформы 11](#_Toc133634948)

[3.3 Обеспечение безопасности приложения. 12](#_Toc133634949)

[4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 15](#_Toc133634951)

[5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ](#_Toc133634952) 18

[ЗАКЛЮЧЕНИЕ 2](#_Toc133634953)1

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 2](#_Toc133634954)2

[ПРИЛОЖЕНИЕ А](#_Toc133634955) 23

[ПРИЛОЖЕНИЕ B 3](#_Toc133634956)6

# ВВЕДЕНИЕ

Сегодня многие люди сталкиваются с проблемой эффективного управления своим временем и задачами. Работа, учеба, личные дела и развлечения могут занимать много времени и быть сложными для организации. Программное средство To-do list может помочь в этом, позволяя пользователям создавать список задач и отслеживать их выполнение.

Программное средство To-do list может помочь пользователям увеличить свою продуктивность, позволяя им оптимизировать свои задачи и распределить своё время более эффективно. Программное средство позволяет пользователям более осознанно подходить к своей работе и учебе, сокращая время на выполнение задач и увеличивая качество работы.

Программное средство может быть полезно для тех, кто хочет улучшить свою организационную способность. Программное средство позволяет пользователям легко управлять своими задачами и дедлайнами, что позволяет им сохранять более четкий и структурированный подход к своей жизни. Это может быть особенное полезно для тех, кто испытывает стресс или чувствует, что их жизнь выходит из-под контроля.

Программное средство является полезным инструментом для всех, кто хочет улучшить свою продуктивность и эффективность в работе и учебе. Оно поможет пользователям более осознанно подходить к своим задачам, улучшить свою организационную способность и уменьшить стресс, связанный с неупорядоченным распределением времени.

Задачей курсового проекта является создание программного средства “To-do list”.

Пояснительная записка разбита на пять глав, в первой из которых рассматриваются аналоги и поставлена задача для проектирования программного средства. Во второй главе описываются основные характеристики и структура ПС. В третьей главе объясняются используемые технологии и технические решения. Четвертый раздел содержит результаты тестирования программного средства, а в пятом разделе приведено руководство пользователя с последовательностью действий для успешного выполнения работы программного средства.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

# **1.1 Обзор аналогов**

В качестве аналога данного программного средства будет рассмотрено веб-приложение todoist.com, предоставляющее услуги task manager. Приложение имеет следующий функционал:

1. регистрация в приложении;
2. вход в аккаунт;
3. добавление новой задачи;
4. просмотр всех созданных задач;
5. группировка задач;
6. установка срока выполнения задачи;
7. установка приоритета для задачи;
8. изменение созданной задачи;
9. удаление созданной задачи;

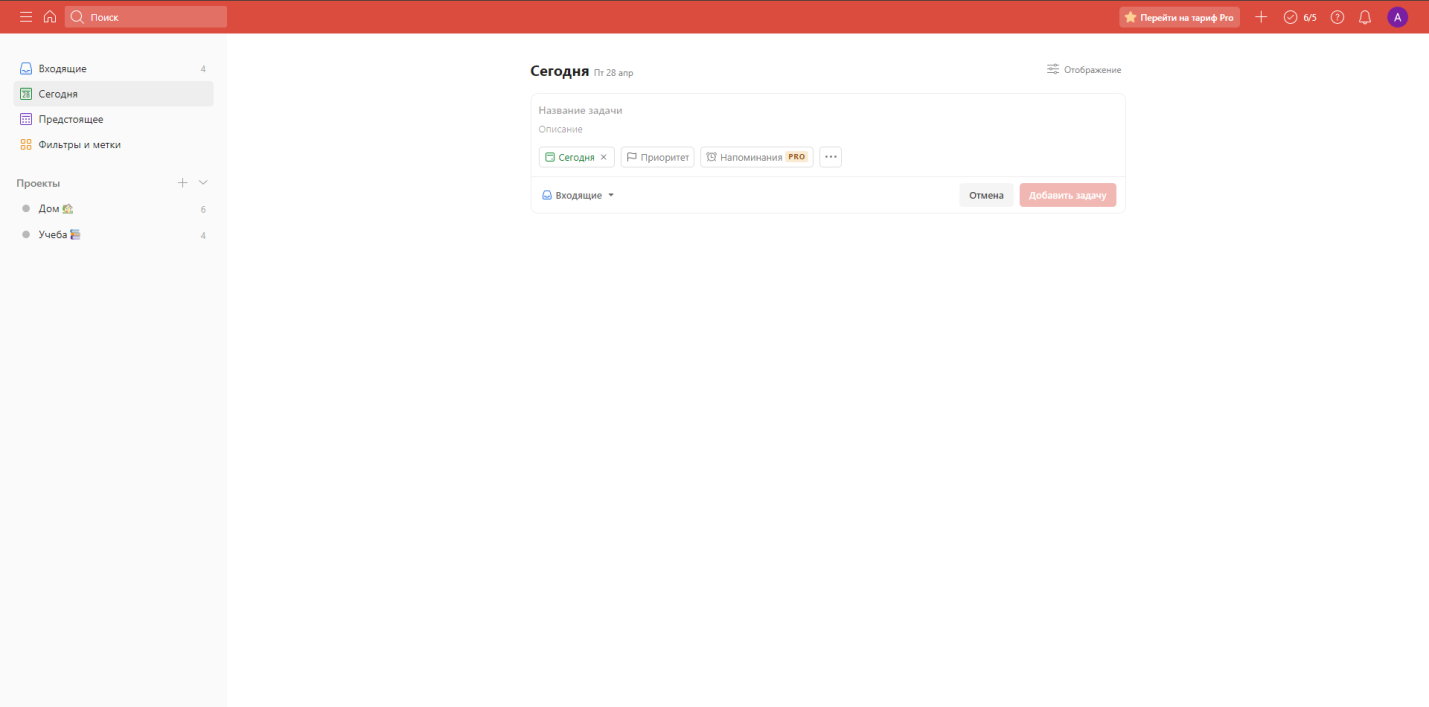


Рисунок 1.1 – Интерфейс главного окна приложения

# **1.2 Формирование требований к проектируемому ПС**

В ходе проведение анализа вышеуказанной информации были сформированы требования для разрабатываемого курсового проекта. В рамках курсового проектирования необходимо реализовать веб-приложение “To-do list”. Приложение должно иметь возможность регистрации и авторизации пользователя, простой и интуитивно понятный пользовательский интерфейс, создание задач, отслеживание статуса этих задач, поиск по всем задачам пользователя, отображение всех созданных пользователем задач, возможность изменения статуса созданных задач, возможность изменения содержимого созданных задач, возможность удаления задач.

Возможность работы с приложением будет иметься только у авторизованных пользователей. Для обеспечения наилучшей безопасности использования следует хранить хеш-код пароля пользователя в базе данных. При создании приложения будет использоваться клиент-серверная архитектура.

# 2 ПРОЕКТИРОВНИЕ ПРОГРАММНОГО СРЕДСТВА

# **2.1 Структура приложения**

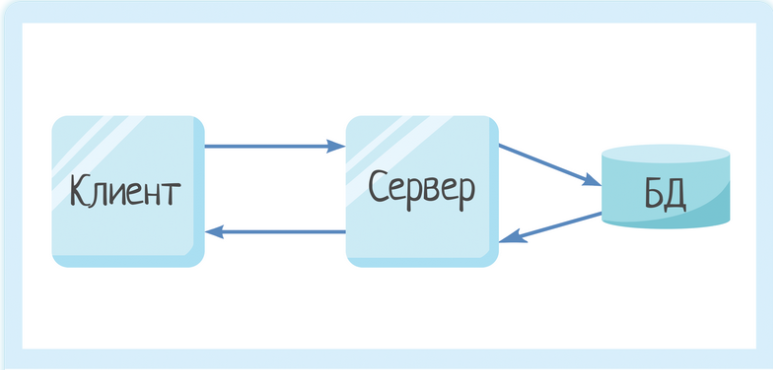


Рисунок 2.1 – Схема работа клиент-серверной архитектуры

Для написания программного средства была выбрана клиент-серверная архитектура. Обе части программного средства строго разделены. Клиентская часть обеспечивает пользовательский интерфейс, реагирует на действия пользователя, выполняя необходимые функции, отправляет при необходимости запросы к серверной части программного средства. Для работы клиентская часть программного средства требует наличия веб-браузера и доступа в интернет.

Серверная часть приложения несет ответственность за бизнес-логику приложения. Сервер получает запросы, на основе которых проводятся необходимые вычисления, формируется ответы, которые посылаются клиенту.

Серверная часть приложения реализована с использованием языка программирования TypeScript, также используется платформа NestJs.

Для хранения и управления данными используется система управления базами данных MongoDB. Благодаря ей осуществляет хранение и извлечение данных, необходимых для работы программного средства.

Реализация клиентской части приложения осуществляется с помощью TypeScript-библиотеки для разработки пользовательских интерфейсов Angular, которая предоставляет высокую скорость разработки, простоту и масштабируемость.

В целом, разделение на клиентскую и серверную части позволяет достичь высокой производительности, безопасности и масштабируемости приложения.

# **2.2 Взаимодействие клиентской и серверной частей**

Взаимодействие клиентской и серверной части было построено с использованием архитектурного стиля REST. REST представляет собой набор правил и принципов, которые определяют, как взаимодействуют клиенты и серверы. Преимуществами использования архитектурного стиля REST являются:

1. Масшатбируемость.
2. Гибкость: REST не зависит от языка программирования, что позволяет использовать различные технологии.
3. Простота: REST использует простой и понятный интерфейс для взаимодействия между клиентом и сервером, что упрощает создание и поддержку веб-сервисов.
4. Надежность: REST поддерживает кэширования, что позволяет уменьшить нагрузку на сервер и ускорить обработку запросов.
5. Безопасность: REST обеспечивает безопасность передачи данных между клиентом и сервером.
6. Поддержка различных форматов данных: REST поддерживает различные форматы данных, такие как JSON, XML и другие, что позволяет клиентам и серверам использовать тот формат данных, который им наиболее удобен.
7. Универсальность: REST может использоваться для создания различных типов веб-сервисов.

В целом, использование архитектурного стиля REST позволяет создавать масштабируемые, гибкие и безопасные веб-сервисы, которые могут обрабатывать большое количество запросов от множества клиентов.

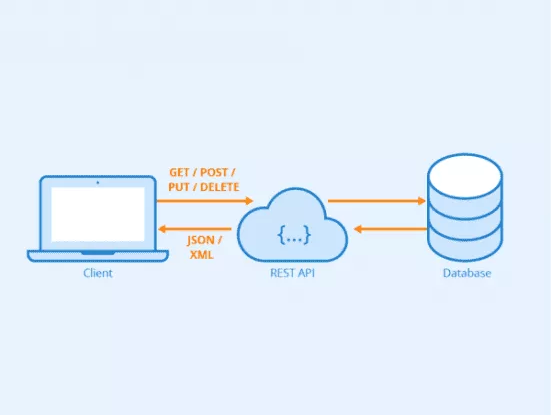


Рисунок 2.2 – Схема работы REST архитектуры

Для передачи данных между клиентом и сервером был выбран формат данных JSON. Преимущества JSON перед другими форматами данных:

1. Легкость: JSON имеет простой и легкий синтаксис, которые позволяет быстро создавать и обрабатывать данные.
2. Простота чтения и записи: JSON-документы легко читаются и записываются человеком, что упрощает отладку и разработку веб-приложений.
3. Поддержка различных языков программирования: JSON может быть использован в различных языках программирования, что позволяет создавать веб-приложения на разных платформах.
4. Более компактный размер: JSON имеет более компактный размер, чем XML, что позволяет уменьшить нагрузку на сеть и ускорить передачу данных.
5. Лучшая производительность: JSON обрабатывается быстрее, чем XML, что позволяет улучшить производительность веб-приложений.
6. Поддержка RESTful API: JSON хорошо подходит для использования RESTful API, так как он позволяет передавать структурированные данные между клиентом и сервером.
7. Широкая поддержка: JSON поддерживается большинством современных браузеров и языков программирования, что делает его универсальным форматов данных для веб-разработки.



Рисунок 2.3 – Сравнение XML и JSON

# **2.3 Проектирование базы данных**

В качестве системы управления базой данных была выбрана система управления базами данных MongoDB. Преимуществами MongoDB является поддержка БД неограниченного размера, мощные и надежные механизмы транзакции и репликации, расширяемая система встроенных языков программирования , наследование, легкая расширяемость.

Для удовлетворения предметной области в базе данных были сформированы следующие отношения: пользователь, задача.

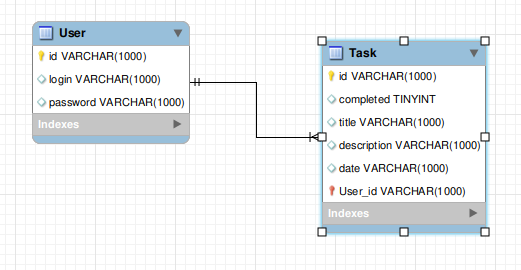


Рисунок 2.4 – Схема базы данных

Между отношения Tasks и Users выставлена связь один ко многим, позволяющая одному пользователь иметь несколько задач.

# 3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

# **3.1 Выбор среды разработки**

Для реализации программного средства использовался текстовый редактор Visual Studio Code. Данный текстовый редактор является оптимальным выбором для разработки с использованием множества языков программирования, в том числе и TypeScript. Благодаря огромному количеству расширений текстовый редактор может предоставлять широкую функциональность для разработки как клиентской, так и серверной части приложения.

# **3.2 Выбор языка программирования и платформы**

Серверная часть приложения была написана на языке TypeScript с использованием платформы NodeJs. NodeJs – это платформа, основанная на движке JavaScript V8, которая позволяет разрабатывать серверные приложения на JavaScript. NodeJs имеет неблокирующую архитектуру, которая позволяет обрабатывать большое количество запросов одновременно без блокировки потока исполнения. NodeJs также имеет обширную библиотеку модулей, которая позволяет быстро и легко добавлять функциональность в приложение. Кроме того, NodeJs предоставляет возможность использовать TypeScript как на серверной стороне, так и на клиентской стороне, что упрощает процесс разработки и позволяет использовать один язык программирования на всех уровнях приложения.

На серверной части используется фрейморк NestJS. Он предоставляет простой и удобный интерфейс для создания серверных приложений, обработки маршрутов и запросов, управления сессиями и шаблонизации. Основными преимуществами NestJS являются минималистичность и гибкость, легкая обработка маршрутов, шаблонизация, управление сессиями, большое количество плагинов и модулей.

Клиентская часть приложения написана с использованием библиотеки Angular. Angular – это библиотека TypeScript, используемая для создания пользовательских интерфейсов. React основан на компонентах, которые позволяют разбить пользовательский интерфейс на множество мелких частей, что упрощает его создание и поддержку. Angular механизм обнаружения изменений(Change Dection, который позволяет минимизировать количество обновлений реальной DOM и увеличивает производительность приложения. Благодаря тому, что React позволяет рендерить компоненты на сервере, ускоряется загрузка страницы. Также для добавления дополнительной функциональности в приложение помогает большое количество плагинов и модулей.

Для обеспечения взаимосвязи между клиентской части и серверной части приложения использовалась библиотека Mongoose. Причинами для выбора Mongoose послужили его простота использования, кроссбраузерность, поддержка Promise, автоматическая сериализация данных, поддержка интерсептеров.

# **3.3 Обеспечение безопасности приложения.**

Для соответствия требованию безопасности приложения была реализована система авторизации и аутентификации. Это необходимо для того, чтобы пользователь имел доступ только к своему аккаунту.

Для реализации аутентификации и авторизации пользователей использовался стандарт для создания токенов доступа JSON Web Token. JWT состоит из трех частей, разделенных точками.

Первой частью является заголовок(Header), который содержит информацию о типе токена и используемом алгоритме шифрования. Обычно это JSON-объект, который содержит два поля: alg – алгоритм шифрования и typ – тип токена.

Вторая часть токена – полезная нагрузка (Payload) – это JSON-объект, который содержит информацию о пользователе или другие данные, связанные с токеном. Полезная нагрузка может содержать любую информацию, которую необходимо передать в приложение.

Третья часть токена – подпись(Signature), являющаяся результатом шифрования заголовка и полезной нагрузки с использованием секретного ключа сервера. Подпись гарантирует целостность токена и защищает его от подделки.

JWT является очень удобным способом передачи информации между клиентом и сервером, так как токен может храниться в локальном хранилище браузера и использоваться для авторизации пользователя на сервере без необходимости повторной аутентификации.

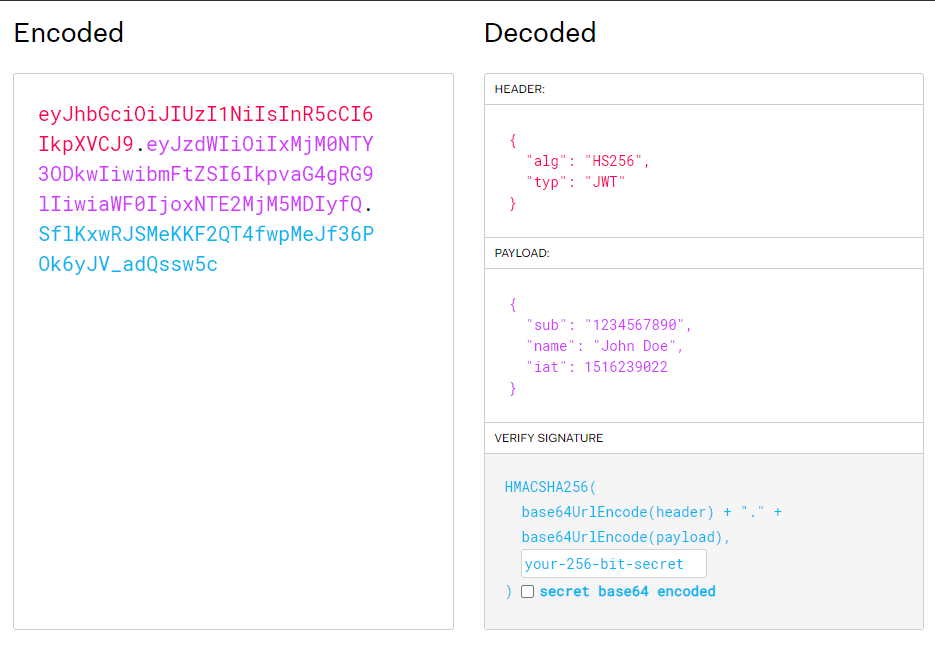


Рисунок 3.1 – Пример JWT.

Для работы с JWT на сервере использовалась библиотека jsonwebtoken.

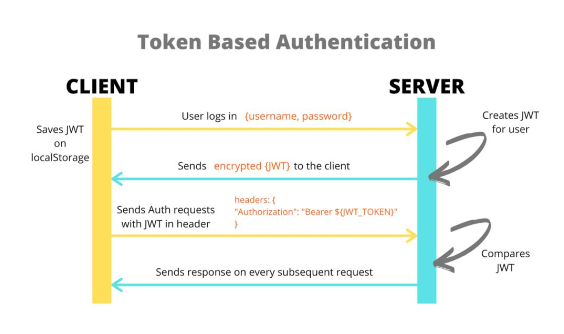


Рисунок 3.2 – Аутентификация с помощью JWT.

Для передачи JWT токена на сервер со стороны клиента в HTTP-пакет при его отправке помещается дополнительный заголовок Authorization, значением которого служит Bearer и сам токен. Для взаимодействия с токеном на сервере написан middleware, который в случае успешной верификации токена передает управление следующему middleware в цепочке. В случае, если токен невалидный, клиент получит ответ со статусом 403.

Для обеспечения безопасности паролей в базе данных помещаются не пароли в открытом виде, а хеш-код пароля. Для генерации хеш-кода на основе пароля используется библиотека bcrypt. Хеш-функция, используемая в bcrypt основана на алгоритме Blowfish. Для обеспечения повышенной безопасности bcrypt реализует хеширование с использованием соли.

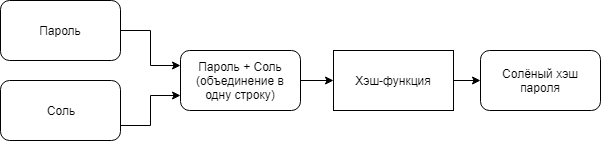


Рисунок 3.3 – Хэширование с использование соли

# 4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

4.1 Тест 1

Таблица 1 – Тест 1

|  |  |
| --- | --- |
| Заглавие | Проверка регистрации |
| Предусловие | Запущенный сервер, открытая страница регистрации |
| Ожидаемый результат | Открытие страницы для авторизации, успешный запрос на сервер |
| Фактический результат |  |

4.2 Тест 2

Таблица 2 – Тест 2

|  |  |
| --- | --- |
| Заглавие | Проверка авторизации |
| Предусловие | Запущенный сервер, открытая страница авторизации |
| Ожидаемый результат | Открытие главной страница, успешный запрос на сервер |
| Фактический результат |  |

4.3 Тест 3

Таблица 3 – Тест 3

|  |  |
| --- | --- |
| Заглавие | Проверка выхода |
| Предусловие | Запущенный сервер |
| Ожидаемый результат | Открытие страницы авторизации |

Продолжение таблицы 3

|  |  |
| --- | --- |
| Фактический результат |  |

4.4 Тест 4

Таблица 4 – Тест 4

|  |  |
| --- | --- |
| Заглавие | Проверка добавления задачи |
| Предусловие | Запущенный сервер, открытая страница для добавления задач, заполненные поля для задачи |
| Ожидаемый результат | Успешный запрос на сервер, отображение задачи на главной страницк |
| Фактический результат |  |

4.5 Тест 5

Таблица 5 – Тест 5

|  |  |
| --- | --- |
| Заглавие | Проверка завершения задачи |
| Предусловие | Запущенный сервер, открытая страница просмотра задачи, нажатие кнопки завершения задачи |
| Ожидаемый результат | Успешный запрос на сервер, смена статуса задачи |
| Фактический результат |  |

4.6 Тест 6

Таблица 6 – Тест 6

|  |  |
| --- | --- |
| Заглавие | Проверка отмены задачи |
| Предусловие | Запущенный сервер, открытая страница просмотра задачи, нажатие кнопки отмены задачи |
| Ожидаемый результат | Успешный запрос на сервер, смена статуса задачи |

|  |  |
| --- | --- |
| Фактический результат |  |

4.7 Тест 7

Таблица 7 – Тест 7

|  |  |
| --- | --- |
| Заглавие | Проверка удаления задачи |
| Предусловие | Запущенный сервер, открытая страница просмотра задачи, нажатие кнопки удаления задачи |
| Ожидаемый результат | Успешный запрос на сервер, смена статуса задачи |
| Фактический результат |  |

# 5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для работы с приложением необходимо перейти по адресу http://localhost:4200.

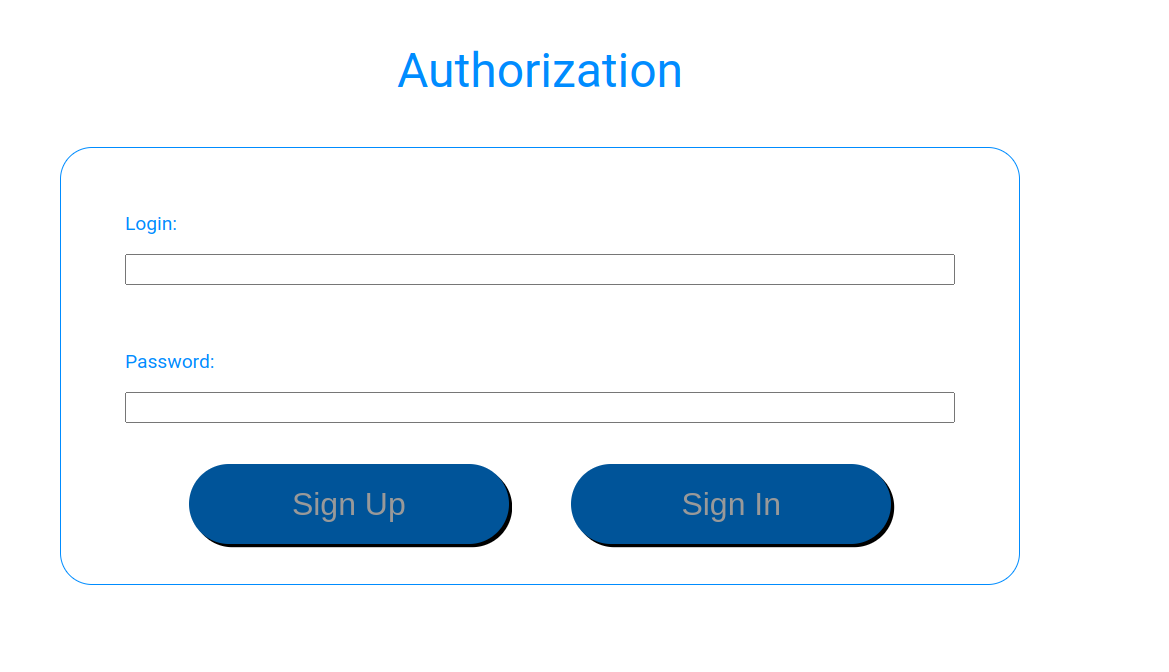


Рисунок 5.1 – Страница авторизации и регистрации

После ввода данных на странице регистрации/автоизации, пользователь попадает на главную страницу приложения.

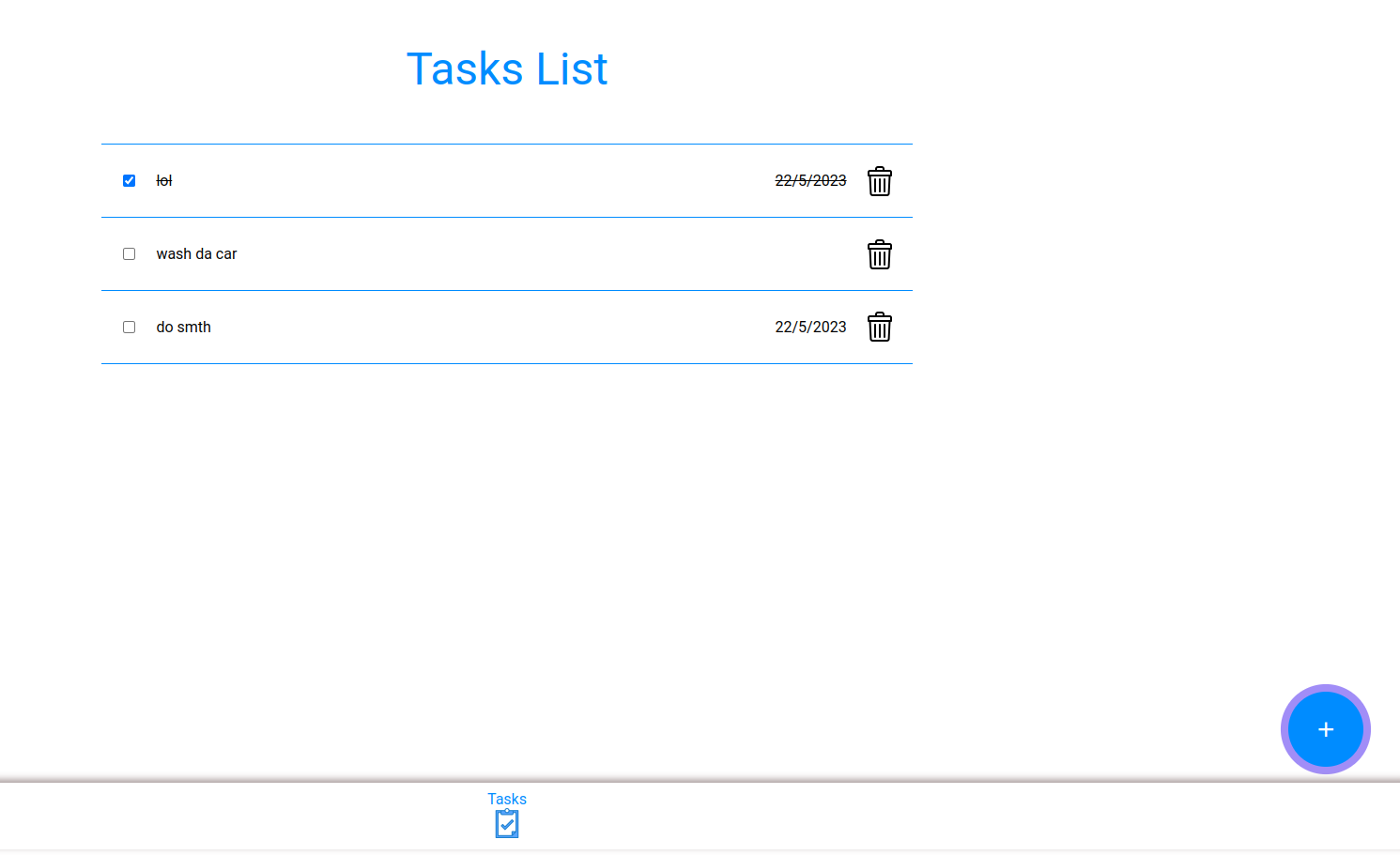


Рисунок 5.2 – Главная страница приложения

На главной странице будет отображены ранее созданные пользователем задачи, кнопка для открытия страница для добавления новой задачи.

При нажатии на кнопку “+” открывается модальное окно, на котором можно добавить новую задачу. Для этого необходимо заполнить все необходимые поля, затем нажать на кнопку “Add”.

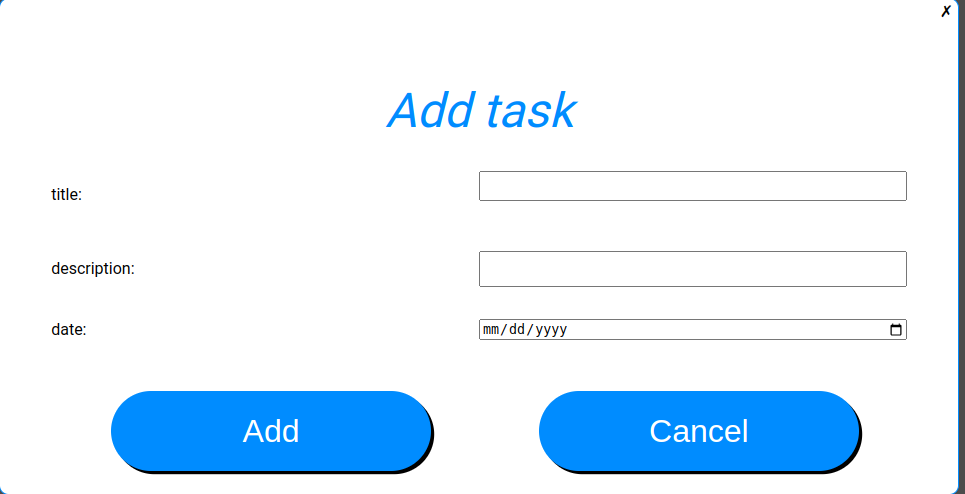


Рисунок 5.3- Страница добавления задачи

После добавления задачи пользователь возвращается на главную страницу. При нажатию на задачу открывается страница просмотра задачи, на которой можно перейти к редактированию задачи.



Рисунок 5.4 – Страница просмотра задачи

# ЗАКЛЮЧЕНИЕ

В рамках выполнения курсового проекта было разработано полноценное веб-приложение, предназначенное для управления задачами.

В процессе работы были изучены платформа NodeJs с использованием NestJS, работа с компьютерными сетями и различные архитектуры веб-приложений.

Ключевое место в выполнении курсового проекта заняло определение архитектуры приложения и используемых функций. В ходе разработки было также проведено тестирование программы. Выявленные в ходе разработки дефекты были устраненные.

Благодаря проведенным исправлениям, удалось создать удобное и простое в использовании веб-приложение “To-do list”, которое позволяет пользователям эффективно управлять своими задачами в режиме онлайн.

Разработка веб-приложения была сложным и многоступенчатым процессом, который требовал определенных знаний в различных сферах информационных технологий. В результате проведенной работы было создано полноценное приложение, которое может быть использовано в повседневной жизни людей для эффективного управления своими задачами.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Cyberforum - [Электронный ресурс]. Режим доступа: <https://www.cyberforum.ru/javascript/>

[2] MDN web docs - [Электронный ресурс]. Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

[3] “HTTP headers” – [Электронный ресурс]. Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

[4] “Введение в REST API — RESTful веб-сервисы” – [Электронный ресурс]. Режим доступа: <https://habr.com/ru/articles/483202/>

[5] “jwt.io” – [Электронный ресурс]. Режим доступа: <https://jwt.io/>

[6] “angular.io” – [Электронный ресурс]. Режим доступа: https://angular.io/

[7] “nestjs.com” – [Электронный ресурс]. Режим доступа: https://nestjs.com/

# ПРИЛОЖЕНИЕ А (рекомендуемое) Исходный код серверной части приложения

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

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

import { NestExpressApplication } from '@nestjs/platform-express';

import mongoose from 'mongoose';

async function bootstrap() {

const app = await NestFactory.create<NestExpressApplication>(AppModule, {

abortOnError: false,

});

app.enableCors();

await app.listen(3000);

await mongoose.connect(

'mongodb+srv://egorsasim:OverTheRainbow228@cluster0.nwipwrh.mongodb.net/to-do-list?authSource=admin&replicaSet=atlas-xu6137-shard-0&w=majority&readPreference=primary&appname=MongoDB%20Compass&retryWrites=true&ssl=true',

);

}

bootstrap();

cat: /home/egorsasim/ToDoListScheduler/server/src/middleware: Is a directory

import { Request, Response, NextFunction } from 'express';

import \* as jwt from 'jsonwebtoken';

export function authMWare(req: Request, res: Response, next: NextFunction) {

const authHeader = req.headers.authorization;

if (!authHeader) {

return res.status(401);

}

const accessToken = authHeader.split(' ')[1];

let userData;

try {

userData = jwt.verify(accessToken, 'randomKey');

} catch {

return res.status(401);

}

req['user'] = userData;

next();

}

cat: /home/egorsasim/ToDoListScheduler/server/src/users: Is a directory

cat: /home/egorsasim/ToDoListScheduler/server/src/users/dto: Is a directory

export class CreateUserDto {

constructor(model: any) {

this.login = model.login;

this.password = model.password;

this.id = model.id;

}

readonly login: string;

readonly password: string;

readonly id: number;

}

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

import { InjectModel } from '@nestjs/mongoose';

import { Model } from 'mongoose';

import { User } from 'src/users/schemas/user.schema';

import \* as bcrypt from 'bcrypt';

import \* as jwt from 'jsonwebtoken';

@Injectable()

export class UsersService {

constructor(@InjectModel(User.name) private userModel: Model<User>) {}

async registrate(login: string, password: string): Promise<string> {

const isLoginExists = await this.userModel.findOne({ login: login });

if (isLoginExists) {

return '409'; //conflict code; cause user already exists;

}

const passwordHash = await bcrypt.hash(password, 3);

const user = await this.userModel.create({

login: login,

password: passwordHash,

});

const accessToken = jwt.sign({ id: user.id, login: login }, 'randomKey', {

expiresIn: '365d',

});

return accessToken;

}

async login(login: string, password: string): Promise<string> {

const user = await this.userModel.findOne({ login: login });

if (!user) {

return '404'; //404 not found; if user with such name not found

}

const isPasswordValid = await bcrypt.compare(password, user.password);

if (!isPasswordValid) {

return '401'; //410 unauthorized; invalid credentials, password;

}

const accessToken = jwt.sign({ id: user.id, login: login }, 'randomKey', {

expiresIn: '365d',

});

return accessToken;

}

}

import { Controller, HttpCode, Post, Req } from '@nestjs/common';

import { UsersService } from 'src/users/users.service';

@Controller('user')

export class UserController {

constructor(private userService: UsersService) {}

@Post('registration')

@HttpCode(200)

async registrate(@Req() req: Request, res: Response): Promise<any> {

const login = req.body['data']['login'];

const password = req.body['data']['password'];

const accessToken = await this.userService.registrate(login, password);

return { token: accessToken };

}

@Post('login')

@HttpCode(200)

async login(@Req() req: Request): Promise<{ token: string }> {

const login = req.body['data']['login'];

const password = req.body['data']['password'];

const accessToken = await this.userService.login(login, password);

return { token: accessToken };

}

}

cat: /home/egorsasim/ToDoListScheduler/server/src/users/schemas: Is a directory

import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';

import { HydratedDocument } from 'mongoose';

export type UserDocument = HydratedDocument<User>;

@Schema()

export class User {

@Prop({ require: true })

login: string;

@Prop({ require: true })

password: string;

}

export const UserSchema = SchemaFactory.createForClass(User);

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

import { UsersService } from './users.service';

import { MongooseModule } from '@nestjs/mongoose';

import { User, UserSchema } from 'src/users/schemas/user.schema';

import { UserController } from 'src/users/user.controller';

@Module({

imports: [

MongooseModule.forFeature([{ name: User.name, schema: UserSchema }]),

],

controllers: [UserController],

providers: [UsersService],

})

export class UsersModule {}

cat: /home/egorsasim/ToDoListScheduler/server/src/tasks: Is a directory

cat: /home/egorsasim/ToDoListScheduler/server/src/tasks/dto: Is a directory

export class CreateTaskDto {

constructor(model: any) {

this.id = model.id;

this.completed = model.completed;

this.title = model.title;

this.description = model.description;

this.date = model.date;

this.user = model.user;

}

readonly id: string;

readonly completed: boolean;

readonly title: string;

readonly description: string;

readonly date: string;

readonly user: string;

}

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

import { InjectModel } from '@nestjs/mongoose';

import { Model } from 'mongoose';

import { CreateTaskDto } from 'src/tasks/dto/create-task.dto';

import { Task } from 'src/tasks/schemas/task.schema';

import { UserTask } from 'src/tasks/tasks.typings';

@Injectable()

export class TasksService {

constructor(@InjectModel(Task.name) private taskModel: Model<Task>) {}

async getTasks(userId: string): Promise<UserTask[]> {

const tasks = await this.taskModel.find({ user: userId });

const tasksDto = tasks.map((task) => {

return new CreateTaskDto(task);

});

return tasksDto;

}

async getTask(id: string): Promise<UserTask> {

const task = await this.taskModel.findById(id);

return new CreateTaskDto(task);

}

async removeTask(id: string): Promise<void> {

await this.taskModel.findByIdAndRemove(id);

}

async addTask(task: UserTask, userId: string): Promise<void> {

await this.taskModel.create({ ...task, user: userId });

}

async updateTasks(task: UserTask): Promise<void> {

await this.taskModel.findByIdAndUpdate(task.id, task);

}

}

export interface UserTask {

id: string;

completed: boolean;

title: string;

description: string;

date: string;

user: string;

}

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

import { TasksController } from 'src/tasks/tasks.controller';

import { TasksService } from 'src/tasks/tasks.service';

import { MongooseModule } from '@nestjs/mongoose';

import { Task, TaskSchema } from 'src/tasks/schemas/task.schema';

import { authMWare } from 'src/middleware/auth';

@Module({

imports: [

MongooseModule.forFeature([{ name: Task.name, schema: TaskSchema }]),

],

controllers: [TasksController],

providers: [TasksService],

})

export class TasksModule {

configure(consumer: MiddlewareConsumer) {

consumer.apply(authMWare).forRoutes(TasksController);

}

}

cat: /home/egorsasim/ToDoListScheduler/server/src/tasks/schemas: Is a directory

import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';

import mongoose, { HydratedDocument } from 'mongoose';

import { User } from 'src/users/schemas/user.schema';

export type TaskDocument = HydratedDocument<Task>;

@Schema()

export class Task {

@Prop({ required: true })

title: string;

@Prop()

description: string;

@Prop()

date: string;

@Prop()

completed: boolean;

@Prop({ type: mongoose.Schema.Types.ObjectId, ref: 'User' })

user: User;

}

export const TaskSchema = SchemaFactory.createForClass(Task);

import {

Controller,

Delete,

Get,

HttpCode,

Param,

Post,

Put,

Req,

} from '@nestjs/common';

import { TasksService } from 'src/tasks/tasks.service';

import { UserTask } from 'src/tasks/tasks.typings';

@Controller('tasks')

export class TasksController {

constructor(private tasksService: TasksService) {}

@Get('get')

@HttpCode(200)

async getTasks(@Req() req: Request): Promise<UserTask[]> {

const userId = req['user']['id'];

const tasks = await this.tasksService.getTasks(userId);

return tasks;

}

@Get(':id')

async getTask(@Param() params: any): Promise<UserTask> {

const task = await this.tasksService.getTask(params['id']);

return task;

}

@Delete('remove')

@HttpCode(200)

async removeTask(@Req() req: Request): Promise<void> {

const id: string = req.body['id'];

await this.tasksService.removeTask(id);

}

@Post('add')

@HttpCode(200)

async addTask(@Req() req: Request): Promise<void> {

const userId = req['user']['id'];

await this.tasksService.addTask(req.body['task'], userId);

return;

}

@Put('update')

@HttpCode(200)

async updateTask(@Req() req: Request): Promise<void> {

await this.tasksService.updateTasks(req.body['task']);

}

}

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

import { MongooseModule } from '@nestjs/mongoose';

import { TasksModule } from 'src/tasks/tasks.module';

import { UsersModule } from 'src/users/users.module';

@Module({

imports: [

TasksModule,

UsersModule,

MongooseModule.forRoot(

'mongodb+srv://egorsasim:OverTheRainbow228@cluster0.nwipwrh.mongodb.net/to-do-list?authSource=admin&replicaSet=atlas-xu6137-shard-0&w=majority&readPreference=primary&appname=MongoDB%20Compass&retryWrites=true&ssl=true',

),

],

})

export class AppModule {}

egorsasim:tmp$ ls

server.txt

egorsasim:tmp$ cat $(find ~/ToDoListScheduler/server/src/ -name "\*") > server.txt

cat: /home/egorsasim/ToDoListScheduler/server/src/: Is a directory

cat: /home/egorsasim/ToDoListScheduler/server/src/middleware: Is a directory

cat: /home/egorsasim/ToDoListScheduler/server/src/users: Is a directory

cat: /home/egorsasim/ToDoListScheduler/server/src/users/dto: Is a directory

cat: /home/egorsasim/ToDoListScheduler/server/src/users/schemas: Is a directory

cat: /home/egorsasim/ToDoListScheduler/server/src/tasks: Is a directory

cat: /home/egorsasim/ToDoListScheduler/server/src/tasks/dto: Is a directory

cat: /home/egorsasim/ToDoListScheduler/server/src/tasks/schemas: Is a directory

egorsasim:tmp$ cat $(find ~/ToDoListScheduler/server/src/ -type f -name "\*") > server.txt

egorsasim:tmp$ cat server.txt

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

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

import { NestExpressApplication } from '@nestjs/platform-express';

import mongoose from 'mongoose';

async function bootstrap() {

const app = await NestFactory.create<NestExpressApplication>(AppModule, {

abortOnError: false,

});

app.enableCors();

await app.listen(3000);

await mongoose.connect(

'mongodb+srv://egorsasim:OverTheRainbow228@cluster0.nwipwrh.mongodb.net/to-do-list?authSource=admin&replicaSet=atlas-xu6137-shard-0&w=majority&readPreference=primary&appname=MongoDB%20Compass&retryWrites=true&ssl=true',

);

}

bootstrap();

import { Request, Response, NextFunction } from 'express';

import \* as jwt from 'jsonwebtoken';

export function authMWare(req: Request, res: Response, next: NextFunction) {

const authHeader = req.headers.authorization;

if (!authHeader) {

return res.status(401);

}

const accessToken = authHeader.split(' ')[1];

let userData;

try {

userData = jwt.verify(accessToken, 'randomKey');

} catch {

return res.status(401);

}

req['user'] = userData;

next();

}

export class CreateUserDto {

constructor(model: any) {

this.login = model.login;

this.password = model.password;

this.id = model.id;

}

readonly login: string;

readonly password: string;

readonly id: number;

}

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

import { InjectModel } from '@nestjs/mongoose';

import { Model } from 'mongoose';

import { User } from 'src/users/schemas/user.schema';

import \* as bcrypt from 'bcrypt';

import \* as jwt from 'jsonwebtoken';

@Injectable()

export class UsersService {

constructor(@InjectModel(User.name) private userModel: Model<User>) {}

async registrate(login: string, password: string): Promise<string> {

const isLoginExists = await this.userModel.findOne({ login: login });

if (isLoginExists) {

return '409'; //conflict code; cause user already exists;

}

const passwordHash = await bcrypt.hash(password, 3);

const user = await this.userModel.create({

login: login,

password: passwordHash,

});

const accessToken = jwt.sign({ id: user.id, login: login }, 'randomKey', {

expiresIn: '365d',

});

return accessToken;

}

async login(login: string, password: string): Promise<string> {

const user = await this.userModel.findOne({ login: login });

if (!user) {

return '404'; //404 not found; if user with such name not found

}

const isPasswordValid = await bcrypt.compare(password, user.password);

if (!isPasswordValid) {

return '401'; //410 unauthorized; invalid credentials, password;

}

const accessToken = jwt.sign({ id: user.id, login: login }, 'randomKey', {

expiresIn: '365d',

});

return accessToken;

}

}

import { Controller, HttpCode, Post, Req } from '@nestjs/common';

import { UsersService } from 'src/users/users.service';

@Controller('user')

export class UserController {

constructor(private userService: UsersService) {}

@Post('registration')

@HttpCode(200)

async registrate(@Req() req: Request, res: Response): Promise<any> {

const login = req.body['data']['login'];

const password = req.body['data']['password'];

const accessToken = await this.userService.registrate(login, password);

return { token: accessToken };

}

@Post('login')

@HttpCode(200)

async login(@Req() req: Request): Promise<{ token: string }> {

const login = req.body['data']['login'];

const password = req.body['data']['password'];

const accessToken = await this.userService.login(login, password);

return { token: accessToken };

}

}

import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';

import { HydratedDocument } from 'mongoose';

export type UserDocument = HydratedDocument<User>;

@Schema()

export class User {

@Prop({ require: true })

login: string;

@Prop({ require: true })

password: string;

}

export const UserSchema = SchemaFactory.createForClass(User);

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

import { UsersService } from './users.service';

import { MongooseModule } from '@nestjs/mongoose';

import { User, UserSchema } from 'src/users/schemas/user.schema';

import { UserController } from 'src/users/user.controller';

@Module({

imports: [

MongooseModule.forFeature([{ name: User.name, schema: UserSchema }]),

],

controllers: [UserController],

providers: [UsersService],

})

export class UsersModule {}

export class CreateTaskDto {

constructor(model: any) {

this.id = model.id;

this.completed = model.completed;

this.title = model.title;

this.description = model.description;

this.date = model.date;

this.user = model.user;

}

readonly id: string;

readonly completed: boolean;

readonly title: string;

readonly description: string;

readonly date: string;

readonly user: string;

}

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

import { InjectModel } from '@nestjs/mongoose';

import { Model } from 'mongoose';

import { CreateTaskDto } from 'src/tasks/dto/create-task.dto';

import { Task } from 'src/tasks/schemas/task.schema';

import { UserTask } from 'src/tasks/tasks.typings';

@Injectable()

export class TasksService {

constructor(@InjectModel(Task.name) private taskModel: Model<Task>) {}

async getTasks(userId: string): Promise<UserTask[]> {

const tasks = await this.taskModel.find({ user: userId });

const tasksDto = tasks.map((task) => {

return new CreateTaskDto(task);

});

return tasksDto;

}

async getTask(id: string): Promise<UserTask> {

const task = await this.taskModel.findById(id);

return new CreateTaskDto(task);

}

async removeTask(id: string): Promise<void> {

await this.taskModel.findByIdAndRemove(id);

}

async addTask(task: UserTask, userId: string): Promise<void> {

await this.taskModel.create({ ...task, user: userId });

}

async updateTasks(task: UserTask): Promise<void> {

await this.taskModel.findByIdAndUpdate(task.id, task);

}

}

export interface UserTask {

id: string;

completed: boolean;

title: string;

description: string;

date: string;

user: string;

}

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

import { TasksController } from 'src/tasks/tasks.controller';

import { TasksService } from 'src/tasks/tasks.service';

import { MongooseModule } from '@nestjs/mongoose';

import { Task, TaskSchema } from 'src/tasks/schemas/task.schema';

import { authMWare } from 'src/middleware/auth';

@Module({

imports: [

MongooseModule.forFeature([{ name: Task.name, schema: TaskSchema }]),

],

controllers: [TasksController],

providers: [TasksService],

})

export class TasksModule {

configure(consumer: MiddlewareConsumer) {

consumer.apply(authMWare).forRoutes(TasksController);

}

}

import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';

import mongoose, { HydratedDocument } from 'mongoose';

import { User } from 'src/users/schemas/user.schema';

export type TaskDocument = HydratedDocument<Task>;

@Schema()

export class Task {

@Prop({ required: true })

title: string;

@Prop()

description: string;

@Prop()

date: string;

@Prop()

completed: boolean;

@Prop({ type: mongoose.Schema.Types.ObjectId, ref: 'User' })

user: User;

}

export const TaskSchema = SchemaFactory.createForClass(Task);

import {

Controller,

Delete,

Get,

HttpCode,

Param,

Post,

Put,

Req,

} from '@nestjs/common';

import { TasksService } from 'src/tasks/tasks.service';

import { UserTask } from 'src/tasks/tasks.typings';

@Controller('tasks')

export class TasksController {

constructor(private tasksService: TasksService) {}

@Get('get')

@HttpCode(200)

async getTasks(@Req() req: Request): Promise<UserTask[]> {

const userId = req['user']['id'];

const tasks = await this.tasksService.getTasks(userId);

return tasks;

}

@Get(':id')

async getTask(@Param() params: any): Promise<UserTask> {

const task = await this.tasksService.getTask(params['id']);

return task;

}

@Delete('remove')

@HttpCode(200)

async removeTask(@Req() req: Request): Promise<void> {

const id: string = req.body['id'];

await this.tasksService.removeTask(id);

}

@Post('add')

@HttpCode(200)

async addTask(@Req() req: Request): Promise<void> {

const userId = req['user']['id'];

await this.tasksService.addTask(req.body['task'], userId);

return;

}

@Put('update')

@HttpCode(200)

async updateTask(@Req() req: Request): Promise<void> {

await this.tasksService.updateTasks(req.body['task']);

}

}

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

import { MongooseModule } from '@nestjs/mongoose';

import { TasksModule } from 'src/tasks/tasks.module';

import { UsersModule } from 'src/users/users.module';

@Module({

imports: [

TasksModule,

UsersModule,

MongooseModule.forRoot(

'mongodb+srv://egorsasim:OverTheRainbow228@cluster0.nwipwrh.mongodb.net/to-do-list?authSource=admin&replicaSet=atlas-xu6137-shard-0&w=majority&readPreference=primary&appname=MongoDB%20Compass&retryWrites=true&ssl=true',

),

],

})

export class AppModule {}

egorsasim:tmp$ xclip -i < server.txt

egorsasim:tmp$ gedit server.txt

egorsasim:tmp$ cat server.txt

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

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

import { NestExpressApplication } from '@nestjs/platform-express';

import mongoose from 'mongoose';

async function bootstrap() {

const app = await NestFactory.create<NestExpressApplication>(AppModule, {

abortOnError: false,

});

app.enableCors();

await app.listen(3000);

await mongoose.connect(

'mongodb+srv://egorsasim:OverTheRainbow228@cluster0.nwipwrh.mongodb.net/to-do-list?authSource=admin&replicaSet=atlas-xu6137-shard-0&w=majority&readPreference=primary&appname=MongoDB%20Compass&retryWrites=true&ssl=true',

);

}

bootstrap();

import { Request, Response, NextFunction } from 'express';

import \* as jwt from 'jsonwebtoken';

export function authMWare(req: Request, res: Response, next: NextFunction) {

const authHeader = req.headers.authorization;

if (!authHeader) {

return res.status(401);

}

const accessToken = authHeader.split(' ')[1];

let userData;

try {

userData = jwt.verify(accessToken, 'randomKey');

} catch {

return res.status(401);

}

req['user'] = userData;

next();

}

export class CreateUserDto {

constructor(model: any) {

this.login = model.login;

this.password = model.password;

this.id = model.id;

}

readonly login: string;

readonly password: string;

readonly id: number;

}

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

import { InjectModel } from '@nestjs/mongoose';

import { Model } from 'mongoose';

import { User } from 'src/users/schemas/user.schema';

import \* as bcrypt from 'bcrypt';

import \* as jwt from 'jsonwebtoken';

@Injectable()

export class UsersService {

constructor(@InjectModel(User.name) private userModel: Model<User>) {}

async registrate(login: string, password: string): Promise<string> {

const isLoginExists = await this.userModel.findOne({ login: login });

if (isLoginExists) {

return '409'; //conflict code; cause user already exists;

}

const passwordHash = await bcrypt.hash(password, 3);

const user = await this.userModel.create({

login: login,

password: passwordHash,

});

const accessToken = jwt.sign({ id: user.id, login: login }, 'randomKey', {

expiresIn: '365d',

});

return accessToken;

}

async login(login: string, password: string): Promise<string> {

const user = await this.userModel.findOne({ login: login });

if (!user) {

return '404'; //404 not found; if user with such name not found

}

const isPasswordValid = await bcrypt.compare(password, user.password);

if (!isPasswordValid) {

return '401'; //410 unauthorized; invalid credentials, password;

}

const accessToken = jwt.sign({ id: user.id, login: login }, 'randomKey', {

expiresIn: '365d',

});

return accessToken;

}

}

import { Controller, HttpCode, Post, Req } from '@nestjs/common';

import { UsersService } from 'src/users/users.service';

@Controller('user')

export class UserController {

constructor(private userService: UsersService) {}

@Post('registration')

@HttpCode(200)

async registrate(@Req() req: Request, res: Response): Promise<any> {

const login = req.body['data']['login'];

const password = req.body['data']['password'];

const accessToken = await this.userService.registrate(login, password);

return { token: accessToken };

}

@Post('login')

@HttpCode(200)

async login(@Req() req: Request): Promise<{ token: string }> {

const login = req.body['data']['login'];

const password = req.body['data']['password'];

const accessToken = await this.userService.login(login, password);

return { token: accessToken };

}

}

import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';

import { HydratedDocument } from 'mongoose';

export type UserDocument = HydratedDocument<User>;

@Schema()

export class User {

@Prop({ require: true })

login: string;

@Prop({ require: true })

password: string;

}

export const UserSchema = SchemaFactory.createForClass(User);

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

import { UsersService } from './users.service';

import { MongooseModule } from '@nestjs/mongoose';

import { User, UserSchema } from 'src/users/schemas/user.schema';

import { UserController } from 'src/users/user.controller';

@Module({

imports: [

MongooseModule.forFeature([{ name: User.name, schema: UserSchema }]),

],

controllers: [UserController],

providers: [UsersService],

})

export class UsersModule {}

export class CreateTaskDto {

constructor(model: any) {

this.id = model.id;

this.completed = model.completed;

this.title = model.title;

this.description = model.description;

this.date = model.date;

this.user = model.user;

}

readonly id: string;

readonly completed: boolean;

readonly title: string;

readonly description: string;

readonly date: string;

readonly user: string;

}

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

import { InjectModel } from '@nestjs/mongoose';

import { Model } from 'mongoose';

import { CreateTaskDto } from 'src/tasks/dto/create-task.dto';

import { Task } from 'src/tasks/schemas/task.schema';

import { UserTask } from 'src/tasks/tasks.typings';

@Injectable()

export class TasksService {

constructor(@InjectModel(Task.name) private taskModel: Model<Task>) {}

async getTasks(userId: string): Promise<UserTask[]> {

const tasks = await this.taskModel.find({ user: userId });

const tasksDto = tasks.map((task) => {

return new CreateTaskDto(task);

});

return tasksDto;

}

async getTask(id: string): Promise<UserTask> {

const task = await this.taskModel.findById(id);

return new CreateTaskDto(task);

}

async removeTask(id: string): Promise<void> {

await this.taskModel.findByIdAndRemove(id);

}

async addTask(task: UserTask, userId: string): Promise<void> {

await this.taskModel.create({ ...task, user: userId });

}

async updateTasks(task: UserTask): Promise<void> {

await this.taskModel.findByIdAndUpdate(task.id, task);

}

}

export interface UserTask {

id: string;

completed: boolean;

title: string;

description: string;

date: string;

user: string;

}

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

import { TasksController } from 'src/tasks/tasks.controller';

import { TasksService } from 'src/tasks/tasks.service';

import { MongooseModule } from '@nestjs/mongoose';

import { Task, TaskSchema } from 'src/tasks/schemas/task.schema';

import { authMWare } from 'src/middleware/auth';

@Module({

imports: [

MongooseModule.forFeature([{ name: Task.name, schema: TaskSchema }]),

],

controllers: [TasksController],

providers: [TasksService],

})

export class TasksModule {

configure(consumer: MiddlewareConsumer) {

consumer.apply(authMWare).forRoutes(TasksController);

}

}

import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';

import mongoose, { HydratedDocument } from 'mongoose';

import { User } from 'src/users/schemas/user.schema';

export type TaskDocument = HydratedDocument<Task>;

@Schema()

export class Task {

@Prop({ required: true })

title: string;

@Prop()

description: string;

@Prop()

date: string;

@Prop()

completed: boolean;

@Prop({ type: mongoose.Schema.Types.ObjectId, ref: 'User' })

user: User;

}

export const TaskSchema = SchemaFactory.createForClass(Task);

import {

Controller,

Delete,

Get,

HttpCode,

Param,

Post,

Put,

Req,

} from '@nestjs/common';

import { TasksService } from 'src/tasks/tasks.service';

import { UserTask } from 'src/tasks/tasks.typings';

@Controller('tasks')

export class TasksController {

constructor(private tasksService: TasksService) {}

@Get('get')

@HttpCode(200)

async getTasks(@Req() req: Request): Promise<UserTask[]> {

const userId = req['user']['id'];

const tasks = await this.tasksService.getTasks(userId);

return tasks;

}

@Get(':id')

async getTask(@Param() params: any): Promise<UserTask> {

const task = await this.tasksService.getTask(params['id']);

return task;

}

@Delete('remove')

@HttpCode(200)

async removeTask(@Req() req: Request): Promise<void> {

const id: string = req.body['id'];

await this.tasksService.removeTask(id);

}

@Post('add')

@HttpCode(200)

async addTask(@Req() req: Request): Promise<void> {

const userId = req['user']['id'];

await this.tasksService.addTask(req.body['task'], userId);

return;

}

@Put('update')

@HttpCode(200)

async updateTask(@Req() req: Request): Promise<void> {

await this.tasksService.updateTasks(req.body['task']);

}

}

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

import { MongooseModule } from '@nestjs/mongoose';

import { TasksModule } from 'src/tasks/tasks.module';

import { UsersModule } from 'src/users/users.module';

@Module({

imports: [

TasksModule,

UsersModule,

MongooseModule.forRoot(

'mongodb+srv://egorsasim:OverTheRainbow228@cluster0.nwipwrh.mongodb.net/to-do-list?authSource=admin&replicaSet=atlas-xu6137-shard-0&w=majority&readPreference=primary&appname=MongoDB%20Compass&retryWrites=true&ssl=true',

),

],

}) export class AppModule {}

# ПРИЛОЖЕНИЕ B (рекомендуемое) Исходный код клиентской части приложения

export const SERVER\_ADDRESS: string = 'http://127.0.0.1:3000';

import { ChangeDetectionStrategy, Component, OnInit } from '@angular/core';

import { FormGroup } from '@angular/forms';

import { AuthorizationPageBuilder } from 'src/app/authorization-page/authorization-page.builder';

import { AuthorizationForm } from 'src/app/authorization-page/authorization-page.typings';

import { AuthenticationService } from 'src/app/common/services/authentication.service';

@Component({

selector: 'app-authorization-page',

templateUrl: './authorization-page.component.html',

styleUrls: ['./authorization-page.component.scss'],

providers: [AuthorizationPageBuilder, AuthenticationService],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class AuthorizationPageComponent implements OnInit {

public formGroup: FormGroup<AuthorizationForm> =

this.authorizationPageBuilder.createForm();

constructor(

private authorizationPageBuilder: AuthorizationPageBuilder,

private authenticationService: AuthenticationService

) {}

public ngOnInit(): void {

this.authenticationService.clearStorageToken();

}

public signUp() {

this.authenticationService.signUp(

this.formGroup.get('login').value,

this.formGroup.get('password').value

);

}

public signIn() {

this.authenticationService.signIn(

this.formGroup.get('login').value,

this.formGroup.get('password').value

);

}

}

import { FormControl } from '@angular/forms';

export interface AuthorizationForm {

login: FormControl<string>;

password: FormControl<string>;

}

import { CommonModule } from '@angular/common';

import { NgModule } from '@angular/core';

import { AuthorizationPageComponent } from 'src/app/authorization-page/authorization-page.component';

import { ButtonModule } from 'src/app/common/buttons/button/button.module';

import { ErrorModule } from 'src/app/common/errors/error/error.module';

import { InputModule } from 'src/app/common/inputs/input/input.module';

@NgModule({

declarations: [AuthorizationPageComponent],

imports: [InputModule, ErrorModule, ButtonModule, ErrorModule, CommonModule],

exports: [AuthorizationPageComponent],

})

export class AuthorizationPageModule {}

import { Injectable } from '@angular/core';

import { FormControl, FormGroup, Validators } from '@angular/forms';

import { AuthorizationForm } from 'src/app/authorization-page/authorization-page.typings';

@Injectable()

export class AuthorizationPageBuilder {

public createForm(): FormGroup<AuthorizationForm> {

return new FormGroup({

login: new FormControl(null, [Validators.required]),

password: new FormControl(null, [Validators.required]),

});

}

}

<header class="authorization-page\_\_header">Authorization</header>

<main class="authorization-page\_\_main">

<div class="authorization-page\_\_input-wrap authorization-page\_\_login">

<label class="authorization-page\_\_input-wrap-label">Login:</label>

<app-input

class="authorization-page\_\_input-wrap-label-input"

[control]="formGroup.get('login')"

>

<div

\*ngIf="

formGroup.get('login').errors &&

(formGroup.get('login').touched || formGroup.get('login').dirty)

"

>

<app-error> Required field</app-error>

</div>

</app-input>

</div>

<div class="authorization-page\_\_input-wrap authorization-page\_\_password">

<label class="authorization-page\_\_input-wrap-label">Password:</label>

<app-input

class="authorization-page\_\_input-wrap-input"

[control]="formGroup.get('password')"

>

<div

\*ngIf="

formGroup.get('password').errors &&

(formGroup.get('password').touched || formGroup.get('password').dirty)

"

>

<app-error> Required field</app-error>

</div>

</app-input>

</div>

<div class="authorization-page\_\_controls">

<app-button

class="authorization-page\_\_controls-button"

[borderRadius]="3"

[content]="'Sign Up'"

[disabled]="formGroup.invalid"

(isClicked)="signUp()"

></app-button>

<app-button

class="authorization-page\_\_controls-button"

[borderRadius]="3"

[content]="'Sign In'"

[disabled]="formGroup.invalid"

(isClicked)="signIn()"

></app-button>

</div>

</main>

.authorization-page\_\_header {

text-align: center;

color: var(--main-font-color);

font-size: 3rem;

padding: 3rem;

}

.authorization-page\_\_main {

width: 50%;

margin: 0 auto;

border: 1px solid var(--main-border-color);

padding: 2.5rem;

border-radius: 2rem;

}

.authorization-page\_\_input-wrap {

padding: 1.5rem;

}

.authorization-page\_\_input-wrap-label {

color: var(--main-font-color);

margin-bottom: 1.2rem;

display: block;

font-size: 1.2rem;

}

.authorization-page\_\_controls {

width: 80%;

display: flex;

margin: 0 auto;

justify-content: space-between;

}

import { Injectable } from '@angular/core';

import {

HttpRequest,

HttpHandler,

HttpEvent,

HttpInterceptor,

} from '@angular/common/http';

import { Observable } from 'rxjs';

import { finalize } from 'rxjs/operators';

import { LoaderService } from 'src/app/common/services/loader.service';

@Injectable()

export class LoadingInterceptor implements HttpInterceptor {

private totalRequests = 0;

constructor(private loaderService: LoaderService) {}

intercept(

request: HttpRequest<unknown>,

next: HttpHandler

): Observable<HttpEvent<unknown>> {

this.totalRequests++;

this.loaderService.setLoading(true);

return next.handle(request).pipe(

finalize(() => {

this.totalRequests--;

if (this.totalRequests == 0) {

this.loaderService.setLoading(false);

}

})

);

}

}

.with-nav-bar\_\_nab-bar {

position: absolute;

bottom: 0.5rem;

width: 100%;

}

import { ChangeDetectionStrategy, Component } from '@angular/core';

@Component({

selector: 'app-with-nav-bar',

templateUrl: './with-nav-bar.component.html',

styleUrls: ['./with-nav-bar.component.scss'],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class WithNavBarComponent {}

import { NgModule } from '@angular/core';

import { AppRoutingModule } from 'src/app/app-routing.module';

import { NavBarModule } from 'src/app/nav-bar/nav-bar.module';

import { WithNavBarComponent } from 'src/app/with-nav-bar/with-nav-bar.component';

@NgModule({

declarations: [WithNavBarComponent],

imports: [AppRoutingModule, NavBarModule],

exports: [WithNavBarComponent],

})

export class WithNavBarModule {}

<router-outlet></router-outlet>

<nav-bar class="with-nav-bar\_\_nab-bar"></nav-bar>

import { NgModule } from '@angular/core';

import { RouterModule } from '@angular/router';

import { WithoutNavBarComponent } from 'src/app/without-nav-bar/without-nav-bar.component';

@NgModule({

declarations: [WithoutNavBarComponent],

imports: [RouterModule],

exports: [WithoutNavBarComponent],

})

export class WithoutNavBarModule {}

<router-outlet></router-outlet>

import { ChangeDetectionStrategy, Component } from '@angular/core';

@Component({

selector: 'without-nav-bar',

templateUrl: './without-nav-bar.component.html',

styleUrls: ['./without-nav-bar.component.scss'],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class WithoutNavBarComponent {}

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { AuthorizationPageComponent } from 'src/app/authorization-page/authorization-page.component';

import { TaskDetailedPageComponent } from 'src/app/common/tasks/tasks-detailed-page/task-detailed-page.component';

import { TasksPageComponent } from 'src/app/tasks/tasks-page/tasks-page.component';

import { WithNavBarComponent } from 'src/app/with-nav-bar/with-nav-bar.component';

import { WithoutNavBarComponent } from 'src/app/without-nav-bar/without-nav-bar.component';

const routes: Routes = [

{

path: '',

pathMatch: 'full',

redirectTo: 'without-nav-bar',

},

{

path: 'with-nav-bar',

component: WithNavBarComponent,

children: [

{

path: '',

pathMatch: 'full',

redirectTo: 'tasks-page',

},

{

path: 'tasks-page',

component: TasksPageComponent,

},

],

},

{

path: 'without-nav-bar',

component: WithoutNavBarComponent,

children: [

{

path: '',

pathMatch: 'full',

redirectTo: 'authorization',

},

{

path: 'authorization',

component: AuthorizationPageComponent,

loadChildren: () =>

import('./authorization-page/authorization-page.module').then(

(m) => m.AuthorizationPageModule

),

},

{

path: 'task-detailed-page/:id',

component: TaskDetailedPageComponent,

loadChildren: () =>

import(

'../app/common/tasks/tasks-detailed-page/task-detailed-page.module'

).then((m) => m.TaskDetailedPageModule),

},

],

},

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule],

})

export class AppRoutingModule {}

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.scss'],

})

export class AppComponent {

title = 'todo-list-scheduler';

}

export const NO\_INET\_ACCESS =

'error occured, server not responding, please, check internet connection';

.input\_\_wrap {

display: flex;

width: 100%;

flex-direction: column;

row-gap: 0.5rem;

overflow: hidden;

}

.input\_\_value {

width: 100%;

flex: 1;

}

.input\_\_error-wrap {

flex: 1;

position: relative;

}

<div

class="input\_\_wrap"

[style.minWidth]="width + 'rem'"

[style.minHeight]="height + 'rem'"

>

<input class="input\_\_value" [type]="type" [formControl]="control" />

<div class="input\_\_error-wrap">

<ng-content class="input\_\_error"></ng-content>

</div>

</div>

export enum InputTypes {

Button = 'button',

Checkbox = 'checkbox',

Color = 'color',

Date = 'date',

DateTimeLocal = 'datetime-local',

Email = 'email',

File = 'file',

Hidden = 'hidden',

Image = 'image',

Month = 'month',

Number = 'number',

Password = 'password',

Radio = 'radio',

Range = 'range',

Reset = 'reset',

Search = 'search',

Submit = 'submit',

Tel = 'tel',

Text = 'text',

Time = 'time',

Url = 'url',

Week = 'week',

}

export interface InputError {

error: string;

description: string;

}

import { CommonModule } from '@angular/common';

import { NgModule } from '@angular/core';

import { ReactiveFormsModule } from '@angular/forms';

import { InputComponent } from 'src/app/common/inputs/input/input.component';

@NgModule({

declarations: [InputComponent],

imports: [ReactiveFormsModule, CommonModule],

exports: [InputComponent],

})

export class InputModule {}

import { ChangeDetectionStrategy, Component, Input } from '@angular/core';

import { FormControl } from '@angular/forms';

import {

InputError,

InputTypes,

} from 'src/app/common/inputs/input/input.constants';

@Component({

selector: 'app-input',

templateUrl: './input.component.html',

styleUrls: ['./input.component.scss'],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class InputComponent {

@Input() public control: FormControl<any>;

@Input() public type: InputTypes = InputTypes.Text;

@Input() public errorsHandlers: InputError[];

@Input() public height: number = 3;

@Input() public width: number = 5;

}

<button

class="button"

[style.width]="width + 'rem'"

[style.height]="height + 'rem'"

[style.fontSize]="fontSize + 'rem'"

[style.color]="contentColor"

[style.backgroundColor]="bgColor"

[style.borderRadius]="borderRadius + 'rem'"

[class.is-disabled]="disabled"

(click)="isClicked.emit()"

>

{{ content }}

</button>

import { NgModule } from '@angular/core';

import { ButtonComponent } from 'src/app/common/buttons/button/button.component';

@NgModule({

declarations: [ButtonComponent],

exports: [ButtonComponent],

})

export class ButtonModule {}

.button {

border: none;

text-decoration: none;

cursor: pointer;

box-shadow: 0.2rem 0.2rem var(--box-shadow-color);

transition: 0.3s;

&:hover {

filter: brightness(120%);

}

&:active {

box-shadow: none;

}

}

.is-disabled {

filter: brightness(60%);

pointer-events: none;

}

import {

ChangeDetectionStrategy,

Component,

EventEmitter,

Input,

Output,

} from '@angular/core';

@Component({

selector: 'app-button',

templateUrl: './button.component.html',

styleUrls: ['./button.component.scss'],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class ButtonComponent {

@Input() public width: number = 20;

@Input() public height: number = 5;

@Input() public fontSize: number = 2;

@Input() public content: string = '';

@Input() public contentColor: string = 'white';

@Input() public bgColor: string = 'rgb(0, 140, 255)';

@Input() public borderRadius: number = 0;

@Input() public disabled: boolean = false;

@Output() public isClicked: EventEmitter<void> = new EventEmitter();

}

import { NgModule } from '@angular/core';

import { PulsingButtonComponent } from 'src/app/common/buttons/pulsing-button/pulsing-button.component';

@NgModule({

declarations: [PulsingButtonComponent],

exports: [PulsingButtonComponent],

})

export class PulsingButtonModule {}

<button

[style.width]="width + 'rem'"

[style.height]="height + 'rem'"

[style.backgroundColor]="backgroundColor"

[style.color]="contentColor"

[style.fontSize]="contentSize + 'rem'"

class="pulsing-button"

(click)="isClicked.emit()"

>

{{ content }}

</button>

.pulsing-button {

display: flex;

align-items: center;

justify-content: center;

color: var(--control-color);

position: relative;

border: none;

box-shadow: 0 0 0 0 rgba(92, 58, 241, 0.7);

border-radius: 50%;

background-color: var(--control-bg-color);

background-size: cover;

background-repeat: no-repeat;

cursor: pointer;

-webkit-animation: pulse 1.25s infinite cubic-bezier(0.66, 0, 0, 1);

-moz-animation: pulse 1.25s infinite cubic-bezier(0.66, 0, 0, 1);

-ms-animation: pulse 1.25s infinite cubic-bezier(0.66, 0, 0, 1);

animation: pulse 1.25s infinite cubic-bezier(0.66, 0, 0, 1);

&:hover {

-webkit-animation: none;

-moz-animation: none;

-ms-animation: none;

animation: none;

}

}

@-webkit-keyframes pulse {

to {

box-shadow: 0 0 0 45px rgba(232, 76, 61, 0);

}

}

@-moz-keyframes pulse {

to {

box-shadow: 0 0 0 45px rgba(232, 76, 61, 0);

}

}

@-ms-keyframes pulse {

to {

box-shadow: 0 0 0 45px rgba(232, 76, 61, 0);

}

}

@keyframes pulse {

to {

box-shadow: 0 0 0 45px rgba(232, 76, 61, 0);

}

}

import {

ChangeDetectionStrategy,

Component,

EventEmitter,

Input,

OnInit,

Output,

} from '@angular/core';

@Component({

selector: 'app-pulsing-button',

templateUrl: './pulsing-button.component.html',

styleUrls: ['./pulsing-button.component.scss'],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class PulsingButtonComponent {

@Input() public width: number = 5;

@Input() public height: number = 5;

@Input() public backgroundColor: string = 'rgb(0, 140, 255)';

@Input() public contentColor: string = 'white';

@Input() public content: string = '';

@Input() public contentSize: number = 5;

@Output() public isClicked: EventEmitter<void> = new EventEmitter();

}

import { ChangeDetectionStrategy, Component, Input } from '@angular/core';

@Component({

selector: 'app-error',

templateUrl: './error.component.html',

styleUrls: ['./error.component.scss'],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class ErrorComponent {

@Input() public color: string = 'red';

}

import { NgModule } from '@angular/core';

import { ErrorComponent } from 'src/app/common/errors/error/error.component';

@NgModule({

declarations: [ErrorComponent],

exports: [ErrorComponent],

})

export class ErrorModule {}

<div class="error" [style.color]="color">

<ng-content></ng-content>

</div>

import { NgModule } from '@angular/core';

import { TasksCategoriesPanelComponent } from 'src/app/common/tasks/tasks-categories-panel/tasks-categories-panel.component';

@NgModule({

declarations: [TasksCategoriesPanelComponent],

exports: [TasksCategoriesPanelComponent],

})

export class TasksCategoriesPanelModule {}

import { ChangeDetectionStrategy, Component } from '@angular/core';

@Component({

selector: 'app-tasks-categories-panel',

templateUrl: './tasks-categories-panel.component.html',

styleUrls: ['./tasks-categories-panel.component.scss'],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class TasksCategoriesPanelComponent {}

import { Injectable } from '@angular/core';

import { FormControl, FormGroup, Validators } from '@angular/forms';

import { TaskForm } from 'src/app/tasks/task/task.typings';

@Injectable()

export class AddTaskFormBuilder {

public create(): FormGroup<TaskForm> {

return new FormGroup({

id: new FormControl(null),

completed: new FormControl(false),

title: new FormControl('', [Validators.required]),

description: new FormControl(''),

date: new FormControl(null),

});

}

}

import {

ChangeDetectionStrategy,

ChangeDetectorRef,

Component,

EventEmitter,

OnInit,

Output,

} from '@angular/core';

import { FormGroup } from '@angular/forms';

import { TaskService } from 'src/app/common/services/tasks.service';

import { AddTaskFormBuilder } from 'src/app/common/tasks/add-task-form/add-task-form.builder';

import { FormData } from 'src/app/common/typings/typings';

import { TaskForm } from 'src/app/tasks/task/task.typings';

@Component({

selector: 'app-add-task-form',

templateUrl: './add-task-form.component.html',

styleUrls: ['./add-task-form.component.scss'],

changeDetection: ChangeDetectionStrategy.OnPush,

providers: [AddTaskFormBuilder],

})

export class AddTaskModalFormComponent {

@Output() public closeModal: EventEmitter<void> = new EventEmitter();

public formGroup: FormGroup<TaskForm> = this.addTaskFormBuilder.create();

constructor(

private addTaskFormBuilder: AddTaskFormBuilder,

private taskService: TaskService,

private changeDetectorRef: ChangeDetectorRef

) {}

public handleAddTaskButton(): void {

this.formGroup.invalid ? this.formGroup.markAllAsTouched() : this.addTask();

}

private addTask(): void {

this.taskService

.addTask(this.formGroup.value as FormData<TaskForm>)

.subscribe((res) => {

this.taskService.tasksListHasChanged$.next();

this.closeModal.emit();

});

}

}

<header class="add-task-form\_\_header">Add task</header>

<main class="add-task-form\_\_inputs-wrap">

<div class="add-task-form\_\_input">

<span class="add-task-form\_\_input-label">title:</span>

<app-input

class="add-task-form\_\_input-value"

type="text"

[control]="formGroup.get('title')"

>

<div

\*ngIf="

formGroup.get('title').errors &&

(formGroup.get('title').touched || formGroup.get('title').dirty)

"

>

<app-error> Required field</app-error>

</div>

</app-input>

</div>

<div class="add-task-form\_\_input">

<span class="add-task-form\_\_input-label">description:</span>

<textarea

class="add-task-form\_\_input-value add-task-form\_\_input-textarea"

type="text"

[formControl]="formGroup.get('description')"

></textarea>

</div>

<div class="add-task-form\_\_input">

<span class="add-task-form\_\_input-label">date:</span>

<input

class="add-task-form\_\_input-value"

type="date"

[formControl]="formGroup.get('date')"

/>

</div>

</main>

<footer class="add-task-form\_\_controls">

<app-button

class="add-task-form\_\_controls-button add-task-form\_\_control-add-task"

[content]="'Add'"

[contentColor]="'white'"

[borderRadius]="3"

(isClicked)="handleAddTaskButton()"

>

</app-button>

<app-button

class="add-task-form\_\_controls-button add-task-form\_\_control-cancel"

[content]="'Cancel'"

[contentColor]="'white'"

[borderRadius]="3"

(isClicked)="closeModal.emit()"

>

</app-button>

</footer>

<div class="add-task-form\_\_close-modal" (click)="closeModal.emit()">

&#x2717;

</div>

import { CommonModule } from '@angular/common';

import { NgModule } from '@angular/core';

import { ReactiveFormsModule } from '@angular/forms';

import { ButtonModule } from 'src/app/common/buttons/button/button.module';

import { AddTaskModalFormComponent } from 'src/app/common/tasks/add-task-form/add-task-form.component';

import { InputModule } from 'src/app/common/inputs/input/input.module';

import { ErrorModule } from 'src/app/common/errors/error/error.module';

@NgModule({

declarations: [AddTaskModalFormComponent],

imports: [

ReactiveFormsModule,

ButtonModule,

CommonModule,

InputModule,

ErrorModule,

],

exports: [AddTaskModalFormComponent],

})

export class AddTaskModalFormModule {}

.add-task-form\_\_header {

font-size: 3rem;

color: var(--main-font-color);

font-style: italic;

text-align: center;

display: block;

padding: 2rem 0;

}

.add-task-form\_\_inputs-wrap {

display: flex;

flex-direction: column;

row-gap: 2rem;

}

.add-task-form\_\_input-label {

text-align: right;

}

.add-task-form\_\_input {

width: 100%;

display: flex;

justify-content: space-between;

align-items: center;

}

.add-task-form\_\_input-value {

width: 50%;

}

.add-task-form\_\_input-textarea {

resize: none;

}

.add-task-form\_\_controls {

width: 100%;

display: flex;

align-items: center;

justify-content: space-around;

padding-top: 3.2rem;

}

.add-task-form\_\_controls-button {

width: 36%;

height: 3.2rem;

border-radius: 0.6rem;

outline: none;

text-decoration: none;

}

.add-task-form\_\_close-modal {

position: absolute;

top: 0.2rem;

right: 0.3rem;

cursor: pointer;

&:hover {

color: var(--close-color);

}

}

import { NgModule } from '@angular/core';

import { ReactiveFormsModule } from '@angular/forms';

import { RouterModule } from '@angular/router';

import { InputModule } from 'src/app/common/inputs/input/input.module';

import { TaskDetailedPageComponent } from 'src/app/common/tasks/tasks-detailed-page/task-detailed-page.component';

import { ButtonModule } from 'src/app/common/buttons/button/button.module';

import { CommonModule } from '@angular/common';

import { ErrorModule } from 'src/app/common/errors/error/error.module';

@NgModule({

declarations: [TaskDetailedPageComponent],

imports: [

ReactiveFormsModule,

InputModule,

ButtonModule,

CommonModule,

ErrorModule,

],

exports: [TaskDetailedPageComponent],

})

export class TaskDetailedPageModule {}

import { Injectable } from '@angular/core';

import { FormControl, FormGroup, Validators } from '@angular/forms';

import { TaskForm } from 'src/app/tasks/task/task.typings';

@Injectable()

export class TaskDetailedPageBuilder {

public createForm(): FormGroup<TaskForm> {

return new FormGroup({

id: new FormControl(null),

completed: new FormControl(null),

title: new FormControl(null, [Validators.required]),

description: new FormControl(null),

date: new FormControl(null),

});

}

}

import {

ChangeDetectionStrategy,

ChangeDetectorRef,

Component,

} from '@angular/core';

import { FormControl, FormGroup } from '@angular/forms';

import { ActivatedRoute, Router } from '@angular/router';

import { TaskService } from 'src/app/common/services/tasks.service';

import { TaskDetailedPageBuilder } from 'src/app/common/tasks/tasks-detailed-page/task-detailed-page.bilder';

import { FormData } from 'src/app/common/typings/typings';

import { TaskForm } from 'src/app/tasks/task/task.typings';

@Component({

selector: 'app-task-detailed-page',

templateUrl: './task-detailed-page.component.html',

styleUrls: ['./task-detailed-page.component.scss'],

providers: [TaskDetailedPageBuilder],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class TaskDetailedPageComponent {

public formGroup: FormGroup<TaskForm> =

this.taskDetailedPageBuilder.createForm();

public taskId: number;

constructor(

private tasksService: TaskService,

private activatedRoute: ActivatedRoute,

private taskDetailedPageBuilder: TaskDetailedPageBuilder,

private router: Router,

private cdr: ChangeDetectorRef

) {

this.activatedRoute.params.subscribe((val) => {

this.taskId = val['id'];

this.tasksService.getTask(this.taskId).subscribe((formGroup) => {

this.formGroup.setValue(formGroup.value as FormData<TaskForm>);

this.cdr.markForCheck();

});

});

}

public saveTask(): void {

this.tasksService

.updateTask(this.formGroup.value as FormData<TaskForm>)

.subscribe();

this.tasksService.tasksListHasChanged$.next();

this.router.navigate(['/with-nav-bar/tasks-page']);

}

public resetTask(): void {

this.tasksService.getTask(this.taskId).subscribe((formGroup) => {

this.formGroup.setValue(formGroup.value as FormData<TaskForm>);

});

}

}

<header class="task-detailed-page\_\_header">

<app-button

[width]="'6'"

[height]="'3'"

[fontSize]="1"

[borderRadius]="'2'"

[content]="'Reset'"

(isClicked)="resetTask()"

></app-button>

<app-button

[width]="'6'"

[height]="'3'"

[fontSize]="1"

[borderRadius]="'2'"

[content]="'Save'"

[disabled]="formGroup.invalid"

(isClicked)="saveTask()"

></app-button>

</header>

<main class="task-detailed-page">

<app-input [control]="formGroup.get('title')">

<div

\*ngIf="

formGroup.get('title').errors &&

(formGroup.get('title').touched || formGroup.get('title').dirty)

"

>

<app-error> Required field</app-error>

</div>

</app-input>

<textarea

class="task-detailed-page\_\_description"

[formControl]="formGroup.get('description')"

></textarea>

<app-input [type]="'date'" [control]="formGroup.get('date')"></app-input>

<div class="task-detailed-page\_\_mark-as-done">

<label class="task-detailed-page\_\_mark-as-done\_\_label"

>mark as done:

</label>

<input type="checkbox" [formControl]="formGroup.get('completed')" />

</div>

</main>

.task-detailed-page\_\_header {

padding: 1.2rem;

display: flex;

justify-content: right;

column-gap: 1.5rem;

}

.task-detailed-page {

width: 50%;

margin: 0 auto;

}

.task-detailed-page\_\_description {

width: 100%;

resize: none;

height: 6rem;

}

.task-detailed-page\_\_mark-as-done {

display: flex;

align-items: center;

column-gap: 1rem;

}

<div \*ngIf="this.loaderService.getLoading()" class="cssload-container">

<div class="cssload-speeding-wheel"></div>

</div>

.cssload-container {

position: fixed;

width: 100%;

left: 0;

right: 0;

top: 0;

bottom: 0;

background-color: rgba(255, 255, 255, 0.7);

z-index: 9999;

}

.cssload-speeding-wheel {

content: "";

display: block;

position: absolute;

left: 48%;

top: 40%;

width: 6.3rem;

height: 6.3rem;

margin: 0 auto;

border: 0.4rem solid rgb(0, 0, 0);

border-radius: 50%;

border-left-color: transparent;

border-right-color: transparent;

animation: cssload-spin 500ms infinite linear;

-o-animation: cssload-spin 500ms infinite linear;

-ms-animation: cssload-spin 500ms infinite linear;

-webkit-animation: cssload-spin 500ms infinite linear;

-moz-animation: cssload-spin 500ms infinite linear;

}

@keyframes cssload-spin {

100% {

transform: rotate(360deg);

transform: rotate(360deg);

}

}

@-o-keyframes cssload-spin {

100% {

-o-transform: rotate(360deg);

transform: rotate(360deg);

}

}

@-ms-keyframes cssload-spin {

100% {

-ms-transform: rotate(360deg);

transform: rotate(360deg);

}

}

@-webkit-keyframes cssload-spin {

100% {

-webkit-transform: rotate(360deg);

transform: rotate(360deg);

}

}

@-moz-keyframes cssload-spin {

100% {

-moz-transform: rotate(360deg);

transform: rotate(360deg);

}

}

import { Component, ViewEncapsulation } from '@angular/core';

import { LoaderService } from 'src/app/common/services/loader.service';

@Component({

selector: 'app-spinner',

templateUrl: './spinner.component.html',

styleUrls: ['./spinner.component.scss'],

encapsulation: ViewEncapsulation.ShadowDom,

})

export class SpinnerComponent {

constructor(public loaderService: LoaderService) {}

}

import {

AbstractControl,

FormArray,

FormControl,

FormGroup,

} from '@angular/forms';

export type FormData<T> = T extends FormControl<infer FormControlData>

? FormControlData

: T extends FormGroup<infer FormGroupData>

? FormData<FormGroupData>

: T extends FormArray<infer FormArrayData>

? FormData<FormArrayData>[]

: T extends { [K in keyof T]: AbstractControl<unknown> }

? { [K in keyof T]: FormData<T[K]> }

: never;

import { Injectable } from '@angular/core';

@Injectable({

providedIn: 'root',

})

export class LoaderService {

private loading: boolean = false;

constructor() {}

setLoading(loading: boolean) {

this.loading = loading;

}

getLoading(): boolean {

return this.loading;

}

}

import { FormArray, FormControl, FormGroup } from '@angular/forms';

import { FormData } from 'src/app/common/typings/typings';

import { TaskForm } from 'src/app/tasks/task/task.typings';

export function convertTaskDataArrayToReactiveFormArray(

tasks: FormData<TaskForm>[]

): FormArray<FormGroup<TaskForm>> {

const formArray = new FormArray([]);

tasks.forEach((task) =>

formArray.push(

new FormGroup({

id: new FormControl(task.id),

completed: new FormControl(task.completed),

title: new FormControl(task.title),

description: new FormControl(task.description),

date: new FormControl(task.date),

})

)

);

return formArray;

}

export function convertTaskDataToReactiveForm(

task: FormData<TaskForm>

): FormGroup<TaskForm> {

return new FormGroup({

id: new FormControl(task.id),

completed: new FormControl(task.completed),

title: new FormControl(task.title),

description: new FormControl(task.description),

date: new FormControl(task.date),

});

}

import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { FormArray, FormControl, FormGroup } from '@angular/forms';

import { Observable, Subject, catchError, map, of, startWith, tap } from 'rxjs';

import { NO\_INET\_ACCESS } from 'src/app/common/constants/internet\_errors.constants';

import { FormData } from 'src/app/common/typings/typings';

import { SERVER\_ADDRESS } from 'src/app/constants/server.constants';

import { TaskForm } from 'src/app/tasks/task/task.typings';

import {

convertTaskDataArrayToReactiveFormArray,

convertTaskDataToReactiveForm,

} from 'src/app/common/services/helpers';

@Injectable({

providedIn: 'root',

})

export class TaskService {

constructor(private httpClient: HttpClient) {}

public tasksListHasChanged$: Subject<void> = new Subject();

public getTasks(): Observable<FormArray<FormGroup<TaskForm>>> {

return this.httpClient

.get<FormData<TaskForm[]>>(SERVER\_ADDRESS + '/tasks/get', {

headers: {

authorization: `Bearer ${localStorage.getItem('accessToken')}`,

},

})

.pipe(

map((tasks: FormData<TaskForm[]>) =>

convertTaskDataArrayToReactiveFormArray(tasks)

),

catchError(() => of(alert(NO\_INET\_ACCESS)))

) as Observable<FormData<TaskForm[]>>;

}

public getTask(id: number): Observable<FormGroup<TaskForm>> {

return this.httpClient

.get<FormData<TaskForm>>(SERVER\_ADDRESS + '/tasks/' + id, {

headers: {

authorization: `Bearer ${localStorage.getItem('accessToken')}`,

},

})

.pipe(

map((task) => convertTaskDataToReactiveForm(task)),

catchError(() => of(alert(NO\_INET\_ACCESS)))

) as Observable<FormGroup<TaskForm>>;

}

public removeTask(id: number) {

return this.httpClient

.delete(SERVER\_ADDRESS + '/tasks/remove', {

headers: {

authorization: `Bearer ${localStorage.getItem('accessToken')}`,

},

body: { id: id },

})

.pipe(catchError(() => of(alert(NO\_INET\_ACCESS))));

}

public addTask(task: FormData<TaskForm>) {

return this.httpClient

.post(

SERVER\_ADDRESS + '/tasks/add',

{

task: task,

},

{

headers: {

authorization: `Bearer ${localStorage.getItem('accessToken')}`,

},

}

)

.pipe(catchError(() => of(alert(NO\_INET\_ACCESS))));

}

public updateTask(task: FormData<TaskForm>) {

return this.httpClient

.put(

SERVER\_ADDRESS + '/tasks/update',

{

task: task,

},

{

headers: {

authorization: `Bearer ${localStorage.getItem('accessToken')}`,

},

}

)

.pipe(catchError(() => of(alert(NO\_INET\_ACCESS))));

}

}

import { HttpClient, HttpErrorResponse } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { Router } from '@angular/router';

import { catchError, of, tap, throwError } from 'rxjs';

import { NO\_INET\_ACCESS } from 'src/app/common/constants/internet\_errors.constants';

import { SERVER\_ADDRESS } from 'src/app/constants/server.constants';

@Injectable()

export class AuthenticationService {

constructor(private httpClient: HttpClient, private router: Router) {}

public signUp(login: string, password: string): void {

this.httpClient

.post(SERVER\_ADDRESS + '/user/registration', {

data: { login: login, password: password },

})

.pipe(catchError(() => of(alert(NO\_INET\_ACCESS))))

.subscribe((data) => {

if (data['token'] == '409') {

alert('user with such name already exists!!!');

return;

}

this.clearStorageToken();

localStorage.setItem('accessToken', data['token']);

this.router.navigate(['/with-nav-bar/tasks-page']);

});

}

public signIn(login: string, password: string): void {

this.httpClient

.post(SERVER\_ADDRESS + '/user/login', {

data: { login: login, password: password },

})

.pipe(catchError(() => of(alert(NO\_INET\_ACCESS))))

.subscribe((data) => {

this.clearStorageToken();

if (data['token'] == '404') {

alert('user not found!!!');

return;

}

if (data['token'] == '401') {

alert('invalid password!!!');

return;

}

localStorage.setItem('accessToken', data['token']);

this.router.navigate(['/with-nav-bar/tasks-page']);

});

}

public clearStorageToken(): void {

localStorage.removeItem('accessToken');

}

}

<app-spinner></app-spinner>

<router-outlet></router-outlet>

.task {

display: flex;

align-items: center;

justify-content: space-between;

padding: 1.2rem;

margin: 0.2rem 0;

column-gap: 1.2rem;

&.is-overdue {

color: var(--error-color);

}

&.is-completed {

text-decoration: line-through;

}

}

.task\_\_completed {

border-radius: 50%;

}

.task\_\_title {

cursor: pointer;

flex: 1;

display: flex;

align-items: center;

justify-content: left;

}

.task\_\_delete {

display: flex;

align-items: center;

justify-content: right;

}

.task\_\_delete-icon {

width: 2rem;

height: 2rem;

cursor: pointer;

}

<form

class="task"

[class.is-overdue]="

formGroup.get('date').value &&

convertStringToDate(formGroup.get('date').value) < currDate

"

[class.is-completed]="formGroup.get('completed').value"

>

<input

type="checkbox"

class="task\_\_completed"

[formControl]="formGroup.get('completed')"

/>

<div

class="task\_\_title"

routerLink="/without-nav-bar/task-detailed-page/{{

formGroup.get('id').value

}}"

(click)="openDetailedTaskModal(formGroup)"

>

{{ formGroup.get("title").value }}

</div>

<div \*ngIf="formGroup.get('date').value" class="task\_\_date">

{{ formatDate(formGroup.get("date").value) }}

</div>

<div class="task\_\_delete">

<img

class="task\_\_delete-icon"

[src]="trashSrc"

alt="trash icon"

(click)="removeTask(formGroup.get('id').value)"

/>

</div>

</form>

import { FormControl } from '@angular/forms';

export interface TaskForm {

id: FormControl<number>;

completed: FormControl<boolean>;

title: FormControl<string>;

description: FormControl<string>;

date: FormControl<string>;

}

export const TRASH\_SRC = '/assets/icons/task/trash.png';

import { CommonModule } from '@angular/common';

import { NgModule } from '@angular/core';

import { ReactiveFormsModule } from '@angular/forms';

import { AppRoutingModule } from 'src/app/app-routing.module';

import { TaskComponent } from 'src/app/tasks/task/task.component';

@NgModule({

declarations: [TaskComponent],

imports: [ReactiveFormsModule, CommonModule, AppRoutingModule],

exports: [TaskComponent],

})

export class TaskModule {}

import {

ChangeDetectionStrategy,

ChangeDetectorRef,

Component,

EventEmitter,

Input,

OnInit,

Output,

} from '@angular/core';

import { FormGroup } from '@angular/forms';

import { take } from 'rxjs';

import { TaskService } from 'src/app/common/services/tasks.service';

import { FormData } from 'src/app/common/typings/typings';

import { TRASH\_SRC } from 'src/app/tasks/task/task.constants';

import { TaskForm } from 'src/app/tasks/task/task.typings';

@Component({

selector: 'app-task',

templateUrl: './task.component.html',

styleUrls: ['./task.component.scss'],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class TaskComponent implements OnInit {

@Input() public formGroup: FormGroup<TaskForm>;

@Output() public taskIsRemoved: EventEmitter<void> = new EventEmitter();

public readonly trashSrc: string = TRASH\_SRC;

public readonly currDate: Date = new Date(

new Date().setDate(new Date().getDate() - 1)

);

constructor(

private taskService: TaskService,

private cdr: ChangeDetectorRef

) {}

public ngOnInit(): void {

this.handleCompletedState();

}

public openDetailedTaskModal(task: FormGroup<TaskForm>): void {}

public removeTask(id: number) {

this.taskService.removeTask(id).subscribe((response) => {

this.taskService.tasksListHasChanged$.next();

});

}

public handleCompletedState(): void {

this.formGroup.get('completed').valueChanges.subscribe(() => {

this.taskService

.updateTask({

...this.formGroup.value,

completed: !this.formGroup.value.completed,

} as FormData<TaskForm>)

.subscribe((res) => {

this.taskService.tasksListHasChanged$.next();

this.cdr.markForCheck();

});

});

}

public formatDate(date: Date | string): string {

const convertedDate = new Date(date);

return `${convertedDate.getDate()}/${

convertedDate.getMonth() + 1

}/${convertedDate.getFullYear()}`;

}

public convertStringToDate(str: string): Date {

return new Date(str);

}

}

<header class="tasks-list\_\_header">Tasks List</header>

<div

\*ngFor="let task of (tasksList$ | async)?.controls"

class="tasks-list\_\_task"

>

<app-task [formGroup]="task" (taskIsRemoved)="getTasks()"></app-task>

</div>

<div

\*ngIf="!(tasksList$ | async)?.controls.length"

class="tasks-list\_\_empty-list"

>

No created tasks, to create a task click on pulsing button below

</div>

import { CommonModule } from '@angular/common';

import { NgModule } from '@angular/core';

import { TaskModule } from 'src/app/tasks/task/task.module';

import { TasksListComponent } from 'src/app/tasks/tasks-list/tasks-list.component';

@NgModule({

declarations: [TasksListComponent],

imports: [TaskModule, CommonModule],

exports: [TasksListComponent],

})

export class TasksListModule {}

import {

ChangeDetectionStrategy,

ChangeDetectorRef,

Component,

Input,

OnInit,

} from '@angular/core';

import { FormArray, FormGroup } from '@angular/forms';

import {

BehaviorSubject,

ReplaySubject,

Subject,

catchError,

of,

startWith,

} from 'rxjs';

import { TaskService } from 'src/app/common/services/tasks.service';

import { TaskForm } from 'src/app/tasks/task/task.typings';

@Component({

selector: 'app-tasks-list',

templateUrl: './tasks-list.component.html',

styleUrls: ['./tasks-list.component.scss'],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class TasksListComponent implements OnInit {

public tasksList$: BehaviorSubject<FormArray<FormGroup<TaskForm>>> =

new BehaviorSubject(new FormArray([]));

constructor(

private changeDetectorRef: ChangeDetectorRef,

private taskService: TaskService

) {}

public ngOnInit(): void {

this.taskService.tasksListHasChanged$.pipe(startWith('')).subscribe(() => {

this.getTasks();

});

}

public getTasks(): void {

this.taskService

.getTasks()

.subscribe((tasksList: FormArray<FormGroup<TaskForm>>) => {

this.tasksList$.next(tasksList);

});

}

}

.tasks-list\_\_header {

text-align: center;

padding: 3.2rem 0;

font-size: 3rem;

color: var(--main-font-color);

}

.tasks-list\_\_task {

border-top: 1px solid var(--task-border-color);

&:last-child {

border-bottom: 1px solid var(--task-border-color);

}

}

.tasks-list\_\_empty-list {

width: 100%;

color: var(--main-font-color);

text-align: center;

font-size: 1.5rem;

}

import {

ChangeDetectionStrategy,

ChangeDetectorRef,

Component,

OnInit,

} from '@angular/core';

import { FormArray, FormGroup } from '@angular/forms';

import { BehaviorSubject, Observable, ReplaySubject, Subject } from 'rxjs';

import { TaskService } from 'src/app/common/services/tasks.service';

import { TaskForm } from 'src/app/tasks/task/task.typings';

// import { AddTaskModalFormComponent } from 'src/app/common/tasks/add-task-modal-form/add-task-modal-form.component';

@Component({

selector: 'app-tasks-page',

templateUrl: './tasks-page.component.html',

styleUrls: ['./tasks-page.component.scss'],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class TasksPageComponent implements OnInit {

public isOpenedAddTaskModal: BehaviorSubject<boolean> = new BehaviorSubject(

false

);

constructor(

private tasksService: TaskService,

private changeDetectorRef: ChangeDetectorRef

) {}

public ngOnInit(): void {}

public openAddTaskModal(): void {

this.isOpenedAddTaskModal.next(true);

}

public closeAddTaskModal(): void {

this.isOpenedAddTaskModal.next(false);

}

}

import { AsyncPipe } from '@angular/common';

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { PulsingButtonModule } from 'src/app/common/buttons/pulsing-button/pulsing-button.module';

import { AddTaskModalFormModule } from 'src/app/common/tasks/add-task-form/add-task-form.module';

import { TasksListModule } from 'src/app/tasks/tasks-list/tasks-list.module';

import { TasksPageComponent } from 'src/app/tasks/tasks-page/tasks-page.component';

@NgModule({

declarations: [TasksPageComponent],

imports: [

TasksListModule,

PulsingButtonModule,

AsyncPipe,

AddTaskModalFormModule,

BrowserModule,

],

exports: [TasksPageComponent],

})

export class TasksPageModule {}

<app-tasks-list class="tasks-page\_\_tasks-list"></app-tasks-list>

<app-pulsing-button

class="tasks-page\_\_add-task"

mat-mini-fab

[content]="'+'"

[contentSize]="2"

(isClicked)="openAddTaskModal()"

></app-pulsing-button>

<ng-container \*ngIf="isOpenedAddTaskModal | async">

<div

class="tasks-page\_\_add-task-page-background"

(click)="closeAddTaskModal()"

></div>

<app-add-task-form

class="tasks-page\_\_add-task-page"

(closeModal)="closeAddTaskModal()"

></app-add-task-form>

</ng-container>

.tasks-page\_\_tasks-list {

display: block;

width: 45%;

margin: 0 auto;

height: 86%;

overflow-x: hidden;

overflow-y: auto;

}

.tasks-page\_\_add-task {

position: absolute;

right: 3rem;

bottom: 6rem;

}

.tasks-page\_\_add-task-page-background {

position: absolute;

width: 100%;

height: 100%;

top: 0;

left: 0;

background-color: var(--modal-wrap-bg-color);

z-index: 2;

}

.tasks-page\_\_add-task-page {

position: absolute;

top: 25%;

left: 25%;

width: 50%;

border: 0.1rem solid var(--task-border-color);

background-color: var(--page-bg-color);

padding: 3.2rem;

border-radius: 0.6rem;

z-index: 3;

}

:host {

--black: black;

}

import { NgModule } from '@angular/core';

import { RouterModule } from '@angular/router';

import { NavBarComponent } from 'src/app/nav-bar/nav-bar.component';

@NgModule({

declarations: [NavBarComponent],

imports: [RouterModule],

exports: [NavBarComponent],

})

export class NavBarModule {}

import { ChangeDetectionStrategy, Component } from '@angular/core';

import {

CALENDAR\_BLACK\_SRC,

CALENDAR\_BLUE\_SRC,

MENU\_SRC,

PROFILE\_BLACK\_SRC,

PROFILE\_BLUE\_SRC,

TASKS\_BLACK\_SRC,

TASKS\_BLUE\_SRC,

} from 'src/app/nav-bar/nav-bar.constants';

import { NavBarItems } from 'src/app/nav-bar/nav-bar.typings';

@Component({

selector: 'nav-bar',

templateUrl: './nav-bar.component.html',

styleUrls: ['./nav-bar.component.scss'],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class NavBarComponent {

public tasksSrc: string = TASKS\_BLUE\_SRC;

public profileSrc: string = PROFILE\_BLACK\_SRC;

public calendarSrc: string = CALENDAR\_BLACK\_SRC;

public menuSrc: string = MENU\_SRC;

public navBarItems: typeof NavBarItems = NavBarItems;

public selectedItem: NavBarItems = NavBarItems.Tasks;

public changeSelectedNavBarItem(navBarItem: NavBarItems): void {

this.setNavBarItemsToDefault();

this.selectedItem = navBarItem;

switch (navBarItem) {

case NavBarItems.Tasks:

this.tasksSrc = TASKS\_BLUE\_SRC;

break;

case NavBarItems.Calendar:

this.calendarSrc = CALENDAR\_BLUE\_SRC;

break;

case NavBarItems.Me:

this.profileSrc = PROFILE\_BLUE\_SRC;

break;

}

}

private setNavBarItemsToDefault(): void {

this.tasksSrc = TASKS\_BLACK\_SRC;

this.profileSrc = PROFILE\_BLACK\_SRC;

this.calendarSrc = CALENDAR\_BLACK\_SRC;

}

}

<div class="nav-bar">

<div

class="nav-bar\_\_item"

(click)="changeSelectedNavBarItem(navBarItems.Tasks)"

[routerLink]="'/with-nav-bar/tasks-page'"

>

<div

class="nav-bar\_\_item-name"

[class.selected-item]="selectedItem == navBarItems.Tasks"

>

Tasks

</div>

<div>

<img class="nav-bar\_\_item-image" [src]="tasksSrc" alt="tasks-image" />

</div>

</div>

</div>

export enum NavBarItems {

Tasks = 'Tasks',

Calendar = 'Calendar',

Me = 'Me',

}

.nav-bar {

width: 100%;

align-items: center;

justify-content: space-around;

display: flex;

box-shadow: 0rem -0.2rem 0.5rem var(--box-shadow-black);

min-height: 3.5rem;

padding: 0.5rem 0;

}

.nav-bar\_\_item {

text-align: center;

cursor: pointer;

}

.nav-bar\_\_item-image {

width: 2rem;

}

.selected-item {

color: var(--selected-item-color);

}

export const TASKS\_BLACK\_SRC = '/assets/icons/nav-bar/tasks-black.png';

export const TASKS\_BLUE\_SRC = '/assets/icons/nav-bar/tasks-blue.png';

export const CALENDAR\_BLACK\_SRC = '/assets/icons/nav-bar/calendar-black.png';

export const CALENDAR\_BLUE\_SRC = '/assets/icons/nav-bar/calendar-blue.png';

export const PROFILE\_BLACK\_SRC = '/assets/icons/nav-bar/profile-black.png';

export const PROFILE\_BLUE\_SRC = '/assets/icons/nav-bar/profile-blue.png';

export const MENU\_SRC = '/assets/icons/nav-bar/menu.png';

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from 'src/app/app-routing.module';

import { TasksPageModule } from 'src/app/tasks/tasks-page/tasks-page.module';

import { WithNavBarModule } from 'src/app/with-nav-bar/with-nav-bar.module';

import { WithoutNavBarModule } from 'src/app/without-nav-bar/without-nav-bar.module';

import { HTTP\_INTERCEPTORS, HttpClientModule } from '@angular/common/http';

import { AppComponent } from './app.component';

import { AuthorizationPageModule } from 'src/app/authorization-page/authorization-page.module';

import { SpinnerComponent } from 'src/app/common/spinner/spinner.component';

import { LoadingInterceptor } from 'src/app/loading.interceptor';

@NgModule({

declarations: [AppComponent, SpinnerComponent],

imports: [

BrowserModule,

AppRoutingModule,

WithNavBarModule,

TasksPageModule,

WithoutNavBarModule,

HttpClientModule,

AuthorizationPageModule,

],

providers: [

{

provide: HTTP\_INTERCEPTORS,

useClass: LoadingInterceptor,

multi: true,

},

],

bootstrap: [AppComponent],

})

export class AppModule {}

import { TestBed } from '@angular/core/testing';

import { RouterTestingModule } from '@angular/router/testing';

import { AppComponent } from './app.component';

describe('AppComponent', () => {

beforeEach(async () => {

await TestBed.configureTestingModule({

imports: [

RouterTestingModule

],

declarations: [

AppComponent

],

}).compileComponents();

});

it('should create the app', () => {

const fixture = TestBed.createComponent(AppComponent);

const app = fixture.componentInstance;

expect(app).toBeTruthy();

});

it(`should have as title 'todo-list-scheduler'`, () => {

const fixture = TestBed.createComponent(AppComponent);

const app = fixture.componentInstance;

expect(app.title).toEqual('todo-list-scheduler');

});

it('should render title', () => {

const fixture = TestBed.createComponent(AppComponent);

fixture.detectChanges();

const compiled = fixture.nativeElement as HTMLElement;

expect(compiled.querySelector('.content span')?.textContent).toContain('todo-list-scheduler app is running!');

});

});