Blog app

Wstęp :

Schemat:

Aplikacja blog

USER --> FE UI ------> RESTFUL API -----> MONGODB

| |

| |

AUTHORIZATION SERVICE

Apka z firebase, tworzysz projekt na stronie firbase

Idziesz do project settings, znajdujesz <>

I wchodzisz. Wybierasz reacta jak apka w react

Tworzysz i masz tam plik konfiguracyjny do firebasa

Config

Do firebase

Potem idziesz do service accounts bedziesz potrzebowal wygenerowac PRIVATE KEY w przyszlosci

Ustawiasz sobie mongoDb atlas, tworzysz baze cluster connection, PAMIETAJ o mogo uri stringu, nazwie usera, i hasle

**Zaczynamy**

Tworzysz react app. Czyli npx create-react-app nazwa-aplikacji --template typescript.

Jak masz ochotę to można dodać settings.json z ustawieniami do edytora VSC itp.

Dajesz dependencies

Npm i react-router-dom REACTSTRAP (bootstrap wew reacta)

Typy do ts : npm i @types/react-router-dom

Usuwasz niepotrzebne pliki z aplikacji, logo, test, css i w index ts i importy z waznych plików.

W indeksie renderujesz tylko fragment jako poóki co placeholder.

I w pliku index.html usuwasz to co nie potrzebne i didajesz linki do google.fonts,(montserrat ja je lubie) font awesome, custom css

Bootstrap.cdn flaty- idziesz do strony www. Szukasz kompozycji flatly i kopiujesz link HTML

I nie muszi ściągać cssa

Wklejasz link do index.html

Kopijesz fonty z google i dajesz link do index.htmla To samo dot. Font awesome cdn

W indexie dajesz styl do body jakis kolor

Iw style wklejasz threedots.com , to takie loadingi, spinnery wcss

Tworzysz w src assets /styles/dots.css kopiujesz cssa ze strony

WAŻNE skopiuje ze strony po lewej smippet kodu: .stage{css wewnatrz} w index.html w stylach

To robisz na tagiem title TO JEST TAKA OGÓLNA KONFIGURACJA FRONTU

**TERAZ IDZIEMY NA BACKEND będzie on w katalogu server**

Npm init -y żeby stworzyć package.json, potem albo kopiujez tsconfig.json od siebie, albo dajesz tsc --init i sam ci się tworzy. Wewnątrz odznaczasz outdir ./build

Tworzysz skrypty w package i instalujesz dependencies.

Npm i express mongoose potem typy npm i @types/express itp

Jam masz .prettier to możesz go wrzucić na FE

Na froncie tworzysz katalog config i plik congig.ts exportujesz go i umieszczasz

*const config = {*

*firebase: {*

*apiKey: 'AIzaSyANCGIyNtsNBb\_08zMbqB6DAC9OsXbhaGw',*

*authDomain: 'blog-e7187.firebaseapp.com',*

*projectId: 'blog-e7187',*

*storageBucket: 'blog-e7187.appspot.com',*

*messagingSenderId: '582931924811',*

*appId: '1:582931924811:web:ee027e2d5b34559741cace'*

*},*

*server: {*

*url: 'http://localhost:1337'*

*}*

*};*

*export default config;*

Kopiujesz z konsoli ze strony projektu firebase i dajesz zmienna do portu na którym będzie BE.

Tworzysz logging.ts i umieszczasz w ni, schemat:

*const DEFAULT\_NAMESPACE = 'Client';*

*const info = (message: any, namespace?: string) => {*

*if (typeof message === 'string') {*

*console.log(`[${getDate()}] [${namespace || DEFAULT\_NAMESPACE}] [INFO] ${message}`);*

*} else {*

*console.log(`[${getDate()}] [${namespace || DEFAULT\_NAMESPACE}] [INFO]`, message);*

*}*

*};*

*const warn = (message: any, namespace?: string) => {*

*if (typeof message === 'string') {*

*console.log(`[${getDate()}] [${namespace || DEFAULT\_NAMESPACE}] [WARN] ${message}`);*

*} else {*

*console.log(`[${getDate()}] [${namespace || DEFAULT\_NAMESPACE}] [WARN]`, message);*

*}*

*};*

*const error = (message: any, namespace?: string) => {*

*if (typeof message === 'string') {*

*console.log(`[${getDate()}] [${namespace || DEFAULT\_NAMESPACE}] [ERROR] ${message}`);*

*} else {*

*console.log(`[${getDate()}] [${namespace || DEFAULT\_NAMESPACE}] [ERROR]`, message);*

*}*

*};*

*const getDate = () => {*

*return new Date().toISOString();*

*};*

*const logging = { info, warn, error };*

*export default logging;*

Info, error, warning. To będą komunikaty wyświetlane w walidacji

To samo robisz na BE kopiujesz config i logging

*const config = {*

*mongo: {*

*url: `mongodb+srv://lukasz:mypassword@cluster0.tqe6aia.mongodb.net/?retryWrites=true&w=majority`*

*},*

*server: {*

*host: 'localhost',*

*port: 1337*

*}*

*};*

*export default config;*

Logging jest taki sam zmieniasz tylko nazwe w nagłówku

*const DEFAULT\_NAMESPACE = 'Server';*

Jak masz to ustawione idziesz na stronę firbase /service accounts, generujesz new private key **JAKO PLIK JSON.** Tworzysz plik serviceAccountKey.json w katalogu config i wklejasz zawartość wygenerowanego i pobranego klucza -> plik json

Wygląda to tak:

*{*

*"type": "service\_account",*

*"project\_id": "blog-e7187",*

*"private\_key\_id": "fd9b3dbb913031cf370368c8e1319debadee4116",*

*"private\_key": "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQC0xBCop5jo3Zmi\nSnjThT5hdh2ErtNJbOhZNa3ENd7qBESsDVDblg2ad0J/G6BfozwYm8A9knmXULNV\nwzyd14HUlOwSqndHlR/XdqwyhsMDX4i6+Nyh8MB8DP4JuQLCazACZBvcx3VIxzkP\noocismdik7PUBaaacw3WCQEGAIN3LL0hmGFvW/SnJ30fp+2uriPJ1xxiEati2lau\nAXu+sEX2KduLeitkzpsI/UHFEXJ0eIqnFxVIdNn87Zuw6ykIuN8VtE3VmwzRDe9e\n9NaTOKrOBOcNCzMBCJTJBuetM2+C0JBSxmbwov4JPOD1c2QCcGxR7ZvJnyyMHhce\ncB2QLIalAgMBAAECggEAAOTBt7Z9xSt0AEmswLikTBuCj92viisVqkJ+3WrBvH7N\nAhqklX04zUdrn90riUiYgVLXSk7K/FN+kOymkTFOEt0KnmdpQqkIIP0P1dBkVwfr\nKfYadud+EIGI3+k0ee4SY9EgMqo1oMaQpevscScCBt4EzlxlkNF2WtAHQWZEx/A4\nh6Iup86+lH0VZxBI+qCkPM1NL6ykhRgFn6usYCjyVwib43YZobHscp/zJBZox88M\nVesNyT3SpUlxe/AfK88ZMZ8OLYoQiiCtWD4VDjycXz8MnG+AFd8j0ZQMLTRBr2oj\nThsdpQ2idtkzBnfXPQIrad9XCBGud4x0tgIHogu30QKBgQDvMD3hWSV8qAkbUcol\nAocuNg8WI3mVIfJgMeQzBL2GU+hkj2zdF5ZhBwv0fT+MUyPCdddn2CeHFp43drrK\n0feq4Krp0KJ6ee2Bk6WDaAsdFGHimNK1q2pIxjpOLLTVljQVU6nBqgzx6pm8GLDk\nuWH3YSOnXuHfz+HrxYVFRseK2QKBgQDBeJ3A1agnDQkZtQ113hTPjimluvnSOwb0\nkyqTe8KBmDuwQjxB3FSyryZG6+IlVcG3GcZ4es+iac7Nat8sciEMrUW7wus0eCTO\nDV2BopFjKtepgzgNwGhElvI07oev8mEQ8QX9zCq4QllibAMtv31ZBEE6yMIFQAiF\nSgcdu24CrQKBgQDM3t9B05+eRDd4pPel96Jd3IPTjJMpP6hg6JrzcA39roURDSBL\nx1ZQyT5eLaFuyg+JEnJsSeKhfKgMDbA05N9kB4edOczRZvod0J8mDMnoIHsvIYLp\nhGdG6jMqnnRNtZfq5Hw0mzj3ROBlEikB4B+fSh7FzyldEKwLzGU4T0W/EQKBgD4Q\nk4ro/6iHZgP5MPg+5i4vTOz9rmbiyLhs/ZuSeDNawssodGwsYn9v8LePtFyFxj8W\n5bSAsVtI1i7PughoFHIco+I4BXwvjp8LapPLsxg9BjI7BiBjOQ/H3KteWej3uGvW\n5jUk6g9yOXp5bAuhAnUsCEDzZkqI4DLiLKY9Gm2BAoGBAI5PcQTlbqTS80p3bDG0\nGMKuofGfijzeb5TgM8SJ/T5ynr/PcHYBHH6CguPIQOLvS6agjhCLMnwg/jfmayT9\nmau3NfwVaG2YhVR2cJyqcgi7/4PKWE/xOQS35lW26AFnsp+BsmfhHZxSkrj2GsFM\nMuApFMUzRSsvDsL0A1qEP65n\n-----END PRIVATE KEY-----\n",*

*"client\_email": "firebase-adminsdk-a0rc3@blog-e7187.iam.gserviceaccount.com",*

*"client\_id": "109624088107291664122",*

*"auth\_uri": "https://accounts.google.com/o/oauth2/auth",*

*"token\_uri": "https://oauth2.googleapis.com/token",*

*"auth\_provider\_x509\_cert\_url": "https://www.googleapis.com/oauth2/v1/certs",*

*"client\_x509\_cert\_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-a0rc3%40blog-e7187.iam.gserviceaccount.com"*

*}*

**Restful API SETUP**

Tworzysz server.ts w src importujesz co potrzebne. Express, mongoose, config, loggong, mongoose

Póżniej npm i firebase-admin w katalogu server. Pamiętaj. I importujesz

Tworzysz server

*const router = express();*

*/\*\* server handling \*/*

*const httpServer = http.createServer(router);*

Łączysz się z firebase-admin

*//\*\*  connect to firebase admin \*/*

*let serviceAccountKey = require('./config/serviceAccountKey.json');*

*firebaseAdmin.initializeApp({*

*credential: firebaseAdmin.credential.cert(serviceAccountKey)*

*});*

Czas na połączenie z bazą danych- może być w osobnym pliku

*/\*\* Connect to mongo database \*/*

*mongoose*

*.connect(config.mongo.url)*

*.then(() => {*

*logging.info('Mongo database connected');*

*})*

*.catch((error) => {*

*logging.error(error);*

*});*

Ustawiasz sobie logging middleware zeby działał

*/\*\* logging middleware \*/*

*router.use((req, res, next) => {*

*logging.info(`METHOD: '${req.method}' - URL: '${req.url}' -IP: '${req.socket.remoteAddress}'`);*

*res.on('finish', () => {*

*logging.info(`METHOD: '${req.method}' - URL: '${req.url}' -IP: '${req.socket.remoteAddress}' - STATUS: '${res.statusCode}`);*

*});*

*next();*

*});*

Teraz parsujesz body może być jak poniżej, a może być npm i body parser

*/\*\* Parse the body \*/*

*router.use(express.urlencoded({ extended: true }));*

*router.use(express.json());*

Api policies- cokolwiek by to znaczyło

*/\*\* api access policies \*/*

*router.use((req, res, next) => {*

*res.header('Access-Control-Allow-Origin', '\*');*

*res.header('Access-Control-Allow-Origin', 'Origin, X-Requested-With, Content-Type, Accept, Authorization');*

*if (req.method == 'OPTIONS') {*

*res.header('Access-Control-Allow-Methods', 'PUT, POST, PATCH, DELETE, GET');*

*return res.status(200).json({});*

*}*

*next();*

*});*

Prosta walidacja

*/\*\*Error handling \*/*

*router.use((req, res, next) => {*

*const error = new Error('not found');*

*return res.status(404).json({*

*message: error.message*

*});*

*});*

I na koniec nasłuchiwanie

*httpServer.listen(config.server.port, () => {*

*logging.info(`Server is running at ${config.server.host}:${config.server.port}... `);*

*});*

next() pozwala na to że kod dalej się wykonuje, be tego dochodzi do danego miejsca i się zatrzymuje

JAK JUŻ Z GRUBSZA JEST SKONFIGUROWANE DO INSTALUJESZ npm i ts-node-dev /src/server.ts - bez tego bedziesz musiał poprawić w package jsonie

TESTOWANKO W POSTMANIE ALBO INSomii

Client side

Zaczynamy frontend

W src tworzysz plik aplication.tsx Importujesz reacta exportujesz i piszesz interface

Tworzysz katalog config i w nim pliki konfiguracji jak są potrzebne.

Tworzysz potrzebne katalogi, components, contex, interface, pages.

W intrface tworzysz potrzebne interfejsy z potrzebnymi danymi.

Masz katalog config i wnim tworzysz plik konfiguracyjny z routami. TO JEST FAJNE

*import IRoute from '../interfaces/route.interface';*

*const authRoutes: IRoute[] = [*

*];*

*const blogRoutes: IRoute[] = [*

*];*

*const mainRoutes: IRoute[] = [*

*];*

*const routes: IRoute[] = [...authRoutes, ...blogRoutes, ...mainRoutes];*

*export default routes;*

Wracasz do index.ts i robisz tak

*root.render(*

*<BrowserRouter>*

*<Application />*

*</BrowserRouter>*

*);*

Wracasz do application tworzysz interfejs jak nie było?

I mapujesz wszystkie routes

*export interface IApplicationProps {}*

*export const Application = (props: IApplicationProps) => {*

*return (*

*<Switch>*

*{routes.map((route, index) => {*

*return <Route key={index} exact={route.exact} path={route.path} render={(routeProps: RouteChildrenProps<any>) => <route.component {...routeProps} />} />;*

*})}*

*</Switch>*

Najpierw robisz ssobie interfejsy w osobnym katalogu z rzeczami które potrzebujez

Np. *export default interface IRoute {*

*path: string;*

*name: string;*

*exact: boolean;*

*auth: boolean;*

*component: any;*

*props?: any;*

*}*

Potem robisz strony które będa potrzebne i importujesz interfejsy do stron

Jak już masz strony z rafcem to w dodajesz ścieżki do configa. Np:

*import { LoginPage } from '../pages/Login';*

*const blogRoutes: IRoute[] = [*

*{*

*path: '/edit',*

*exact: true,*

*auth: true,*

*component: EditPage,*

*name: 'Edit'*

*},*

*const authRoutes: IRoute[] = [*

*{*

*path: '/login',*

*exact: true,*

*auth: false,*

*component: LoginPage,*

*name: 'Login'*

*},*

*const routes: IRoute[] = [...authRoutes, ...blogRoutes, ...mainRoutes];*

*export default routes;*

Jak masz już podstawowe strony to dodj je do routes w configu

*import IRoute from '../interfaces/route.interface';*

*import { BlogPage } from '../pages/Blog';*

*import { EditPage } from '../pages/Edit';*

*import { HomePage } from '../pages/Home';*

*import { LoginPage } from '../pages/Login';*

*const authRoutes: IRoute[] = [*

*{*

*path: '/login',*

*exact: true,*

*auth: false,*

*component: LoginPage,*

*name: 'Login'*

*},*

*{*

*path: '/register',*

*exact: true,*

*auth: false,*

*component: LoginPage,*

*name: 'Register'*

*}*

*];*

*const blogRoutes: IRoute[] = [*

*{*

*path: '/edit',*

*exact: true,*

*auth: true,*

*component: EditPage,*

*name: 'Edit'*

*},*

*{*

*path: '/edit/:blogID',*

*exact: true,*

*auth: true,*

*component: EditPage,*

*name: 'Edit'*

*},*

*{*

*path: '/blogs/:blogID',*

*exact: true,*

*auth: false,*

*component: BlogPage,*

*name: 'Blog'*

*}*

*];*

*const mainRoutes: IRoute[] = [*

*{*

*path: '/',*

*exact: true,*

*auth: false,*

*component: HomePage,*

*name: 'Home'*

*}*

*];*

*const routes: IRoute[] = [...authRoutes, ...blogRoutes, ...mainRoutes];*

*export default routes;*

Jak już masz skonfigurowane rzeczy to zaczynamy robić komponenty.

Najpierw navbar. Tworzysz plik importujesz reacta i ustaiwasz interface props. Może być pusty.

Buduejsz komponent.

*import React from 'react';*

*import { Link } from 'react-router-dom';*

*import { Navbar, NavbarBrand, Nav, Container } from 'reactstrap';*

*export interface INavbarProps {}*

*export const Navigation = (props: INavbarProps) => {*

*return (*

*<Navbar color="light" light sticky="top" expoand="md">*

*<Container>*

*<NavbarBrand tag={Link} to="/">*

*</NavbarBrand>*

*<Nav className="mr-auto" navbar />*

*</Container>*

*</Navbar>*

*);*

*};*

Budujez kolejny np header.

Ustawiasz interface np taki:

*export interface IHeaderProps {*

*height?: string;*

*image?: string;*

*title: string;*

*headline: string;*

*}*

I tworzysz strukture

Jak utworzysz to dajesz headera do strony głównej home page komponent i tam ustalasz kolejność, strukture

Po sutworzeniu

*import React from 'react';*

*import { Col, Container, Row } from 'reactstrap';*

*export interface IHeaderProps {*

*height?: string;*

*image?: string;*

*title: string;*

*headline: string;*

*children?: any; //add*

*}*

*export const Header = (props: IHeaderProps) => {*

*const { children, height, image, headline, title } = props;*

*let headerStyle = {*

*background: 'linear-gradient(rgba(36, 20, 38, 0.5), rgba(36, 39, 38, 0.5)), url(' + image + ') no-repeat center center',*

*WebkitBackgroundSize: 'cover',*

*MozBackgroundSize: 'cover',*

*OBackgroundSize: 'cover',*

*backgroundSize: 'cover',*

*backgroundRepeat: 'no-repeat',*

*backgroundPosition: 'center',*

*width: '100%',*

*height: height*

*};*

*/\*\* copied and pasted from a webside because i think its nice\*\*/*

*return (*

*<header style={headerStyle}>*

*<Container>*

*<Row className="align-items-center text-center">*

*<Col>*

*<h1 className="display-4 text-white mt-5 mb-2">{title}</h1>*

*<h3 className="mb-5 text-white">{headline}</h3>*

*{children}*

*</Col>*

*</Row>*

*</Container>*

*</header>*

*);*

*};*

*Header.defaultProps = {*

*height: '100%',*

*image: 'https://images.unsplash.com/photo-1488998427799-e3362cec87c3?ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&ixlib=rb-1.2.1&auto=format&fit=crop&w=1950&q=80'*

*};*

Jak już masz headera i nava to idzidziez do homePage i masz coś takiego:

W porpsach importujesz IPage z danymi

*import React from 'react';*

*import { Container } from 'reactstrap';*

*import { Header } from '../components/header/Header';*

*import { Navigation } from '../components/navigation/Navbar';*

*import IPage from '../interfaces/page.interface';*

*export const HomePage = (props: IPage) => {*

*return (*

*<Container fluid className="p-0">*

*<Navigation />*

*<Header title="Nerdy blog side..." headline="you maust check it out!!" />*

*<Container className="mt-5">Blog stuff and things lke that....</Container>*

*</Container>*

*);*

*};*

Można sobie stworzyć puste interfejsy i dopisywać do nich rzeczy które wyjdą w przyszłości że mogą być potrzebne.

Autoryzacja po stronie clienta. FE

Tworzysz plik firebase.ts w katalogu config

*import firebase from 'firebase/compat/app';*

*import 'firebase/compat/auth';*

*import 'firebase/compat/firestore';*

*import config from '../config/config';*

*const Firebase = firebase.initializeApp(config.firebase);*

*export const Providers = {*

*google: new firebase.auth.GoogleAuthProvider()*

*};*

*export const auth = firebase.auth();*

*export default Firebase;*

Potem tworzysz w katalogu models plik auth.ts i tworzysz strukture logowań.

*import firebase from 'firebase/compat';*

*import { auth } from '../config/firebase';*

*export const SignInWithSocialMedia = (provider: firebase.auth.AuthProvider) =>*

*new Promise<firebase.auth.UserCredential>((resolve, reject) => {*

*auth*

*.signInWithPopup(provider)*

*.then((result) => resolve(result))*

*.catch((error) => reject(error));*

*});*

Potem tworzysz konteks teorzysz plik user context.

Dorabiasz w interfejsach user interface z danymi:

*export default interface IUser {*

*\_id: string;*

*uid: string;*

*name: string;*

*}*

*export const DEFAULT\_USER: IUser = {*

*\_id: '',*

*uid: '',*

*name: ''*

*}*

*export const DEFAULT\_FIRE\_TOKEN = '';*

Tworzysz strukture pliku user.context.ts

*import { Dispatch } from 'react';*

*import IUser, {*

*DEFAULT\_FIRE\_TOKEN,*

*DEFAULT\_USER,*

*} from '../interfaces/user.interface';*

*import { createContext } from 'react';*

*export interface IUserState {*

*user: IUser;*

*fire\_token: string;*

*}*

*export interface IUserActions {*

*type: 'login' | 'logout';*

*payload: IUserState;*

*}*

*export const initialUserState: IUserState = {*

*user: DEFAULT\_USER,*

*fire\_token: DEFAULT\_FIRE\_TOKEN,*

*};*

*export const userReducer = (state: IUserState, action: IUserActions) => {*

*let user = action.payload.user;*

*let fire\_token = action.payload.fire\_token;*

*switch (action.type) {*

*case 'login':*

*localStorage.setItem('fire\_token', fire\_token);*

*return { user, fire\_token };*

*case 'logout':*

*localStorage.removeItem('fire\_token');*

*return initialUserState;*

*default:*

*return state;*

*}*

*};*

*export interface IUserContextProps {*

*userState: IUserState;*

*userDispatch: Dispatch<IUserActions>;*

*}*

*const UserContext = createContext<IUserContextProps>({*

*userState: initialUserState,*

*userDispatch: () => {},*

*});*

*export const UserContextConsumer = UserContext.Consumer;*

*export const UserContextProvider = UserContext.Provider;*

*export default UserContext;*

Uzupełniasz application.tsx

*import React, { useEffect, useReducer, useState } from 'react';*

*import { Route, RouteChildrenProps, Switch } from 'react-router-dom';*

*import routes from './config/routes';*

*import { initialUserState, userReducer } from './context/user.context';*

*export interface IApplicationProps {}*

*export const Application = (props: IApplicationProps) => {*

*const [userState, userDispatch] = useReducer(userReducer, initialUserState);*

*const [loading, setLoading] = useState<boolean>(true);*

*/\*\* Used for debugging \*/*

*const [authState, setAuthState] = useState<string>(*

*'Checking local storage...'*

*);*

*useEffect(() => {*

*setTimeout(() => {*

*checkLocalStorageForCredentials();*

*}, 1000);*

*}, []);*

*/\*\* check to see if we have token.*

*\* if we do, verify it with BE*

*\* if not we are logged out initially*

*\*/*

*const checkLocalStorageForCredentials = () => {*

*setAuthState('checking credentials....');*

*const fire\_token = localStorage.getItem('fire\_token');*

*if (fire\_token === null) {*

*userDispatch({ type: 'logout', payload: initialUserState });*

*setAuthState('No credentials found.');*

*setTimeout(() => {*

*setLoading(false);*

*}, 1000);*

*} else {*

*/\*\* validate with BE \*/*

*setAuthState('Credentials found, validating...');*

*setTimeout(() => {*

*setLoading(false);*

*}, 1000);*

*}*

*};*

*const userContextValues = {*

*userState,*

*userDispatch,*

*};*

*if (loading) {*

*}*

*return (*

*<Switch>*

*{routes.map((route, index) => {*

*return (*

*<Route*

*key={index}*

*exact={route.exact}*

*path={route.path}*

*render={(routeProps: RouteChildrenProps<any>) => (*

*<route.component {...routeProps} />*

*)}*

*/>*

*);*

*})}*

*</Switch>*

*);*

*};*

Importujesz spinerka do indexu.tsx

*import './assets/styles/dots.css';*

Jak to mamay to tworzymy kolejny komponent. Loadingcomponent.tsx Czyli importujesz reacta, exportujesz interfejs. I tworzysz strukture

*import React from 'react';*

*import { Card, CardBody } from 'reactstrap';*

*export interface ILoadingProps {*

*dotType?: string;*

*children?: any; //add*

*}*

*export const Loading = (props: ILoadingProps) => {*

*const { children, dotType } = props;*

*return (*

*<div className="text-center">*

*<div className="stage">*

*<div className={dotType}></div>*

*</div>*

*{children}*

*</div>*

*);*

*};*

*Loading.defaultProps = {*

*dotType: 'dot-bricks',*

*};*

*export interface ILoadingComponentProps {*

*card?: boolean;*

*dotType?: string;*

*children?: any; //add*

*}*

*export const LoadingComponent = (props: ILoadingComponentProps) => {*

*const { card, children, dotType } = props;*

*if (card) {*

*return (*

*<Card>*

*<CardBody>*

*<Loading dotType={dotType}>{children}</Loading>*

*</CardBody>*

*</Card>*

*);*

*}*

*return <Loading dotType={dotType}>{children}</Loading>;*

*};*

*LoadingComponent.defaultProps = {*

*card: true,*

*dotType: 'dot-bricks',*

*};*

*export default LoadingComponent;*

Tworzysz kolejny komponent CenterPiece.tsx

*import React from 'react';*

*import { Container } from 'reactstrap';*

*export interface ICenterPieceProps {*

*children?: any; //add*

*}*

*export const CenterPiece = (props: ICenterPieceProps) => {*

*const { children } = props;*

*return (*

*<Container fluid className="p-0">*

*<Container*

*style={{*

*position: 'absolute',*

*left: '50%',*

*top: '50%',*

*transform: 'translate(-50%, -50%)',*

*WebkitTransform: 'translate(-50%, -50%)',*

*}}*

*className="d-flex justify-content-center"*

*>*

*{children}*

*</Container>*

*</Container>*

*);*

*};*

Dodajesz do pliku application.tsx brakujace elementy: Wersja finalna:

*import React, { useEffect, useReducer, useState } from 'react';*

*import { Route, RouteChildrenProps, Switch } from 'react-router-dom';*

*import { AuthRoute } from './components/AuthRoute/AuthRoute';*

*import LoadingComponent from './components/LoadingComponent/Loading';*

*import routes from './config/routes';*

*import {*

*initialUserState,*

*UserContextProvider,*

*userReducer,*

*} from './context/user.context';*

*export interface IApplicationProps {}*

*export const Application = (props: IApplicationProps) => {*

*const [userState, userDispatch] = useReducer(userReducer, initialUserState);*

*const [loading, setLoading] = useState<boolean>(true);*

*/\*\* Used for debugging \*/*

*const [authState, setAuthState] = useState<string>(*

*'Checking local storage...'*

*);*

*useEffect(() => {*

*setTimeout(() => {*

*checkLocalStorageForCredentials();*

*}, 1000);*

*}, []);*

*/\*\* check to see if we have token.*

*\* if we do, verify it with BE*

*\* if not we are logged out initially*

*\*/*

*const checkLocalStorageForCredentials = () => {*

*setAuthState('checking credentials....');*

*const fire\_token = localStorage.getItem('fire\_token');*

*if (fire\_token === null) {*

*userDispatch({ type: 'logout', payload: initialUserState });*

*setAuthState('No credentials found.');*

*setTimeout(() => {*

*setLoading(false);*

*}, 1000);*

*} else {*

*/\*\* validate with BE \*/*

*setAuthState('Credentials found, validating...');*

*setTimeout(() => {*

*setLoading(false);*

*}, 1000);*

*}*

*};*

*const userContextValues = {*

*userState,*

*userDispatch,*

*};*

*if (loading) {*

*<LoadingComponent>{authState}</LoadingComponent>;*

*}*

*return (*

*<UserContextProvider value={userContextValues}>*

*<Switch>*

*{routes.map((route, index) => {*

*if (route.auth) {*

*<Route*

*key={index}*

*exact={route.exact}*

*path={route.path}*

*render={(routeProps: RouteChildrenProps<any>) => (*

*<AuthRoute>*

*<route.component {...routeProps} />*

*</AuthRoute>*

*)}            />;*

*}*

*return (*

*<Route*

*key={index}*

*exact={route.exact}*

*path={route.path}*

*render={(routeProps: RouteChildrenProps<any>) => (*

*<route.component {...routeProps} />*

*)}*

*/>*

*);*

*})}*

*</Switch>*

*</UserContextProvider>*

*);*

*};*

Zabieramy się za Login Page: wersja finalna

*import React, { useContext, useState } from 'react';*

*import { useHistory } from 'react-router-dom';*

*import UserContext from '../context/user.context';*

*import IPage from '../interfaces/page.interface';*

*import firebase from 'firebase/compat';*

*import { SignInWithSocialMedia as SocialMediaPopup } from '../modules/auth';*

*import logging from '../config/logging';*

*import { setEnvironmentData } from 'worker\_threads';*

*import { CenterPiece } from '../components/CenterPiece/CenterPiece';*

*import { Button, Card, CardBody, CardHeader } from 'reactstrap';*

*import { ErrorText } from '../components/ErrorText/ErrorText';*

*import { Providers } from '../config/firebase';*

*import LoadingComponent from '../components/LoadingComponent/Loading';*

*export const LoginPage = (props: IPage) => {*

*const [authenticating, setAuthenticating] = useState<boolean>(false);*

*const [error, setError] = useState<string>('');*

*const userContext = useContext(UserContext);*

*const history = useHistory();*

*const isLogin = window.location.pathname.includes('login');*

*const SignInWithSocialMedia = (provider: firebase.auth.AuthProvider) => {*

*if (error !== '') setError('');*

*setAuthenticating(true);*

*SocialMediaPopup(provider)*

*.then(async (result) => {*

*logging.info(result);*

*let user = result.user;*

*if (user) {*

*let uid = user.uid;*

*let name = user.displayName;*

*if (name) {*

*try {*

*let fire\_token = await user.getIdToken();*

*/\*\* if we get token, auth with BE \*/*

*} catch (error) {*

*setError('Invalid token');*

*logging.error(error);*

*setAuthenticating(false);*

*}*

*} else {*

*/\*\* if no name returned, we could have a custom form here getting users name*

*\* depending on the porvider you are using. google generally return*

*\* ones, lets just use that for now*

*\*/*

*setError('The identity provider doesnt have username');*

*setAuthenticating(false);*

*}*

*} else {*

*setError('Identity provider is missing necessary infrmation');*

*setAuthenticating(false);*

*}*

*})*

*.catch((error) => {*

*setError(error.message);*

*setAuthenticating(false);*

*});*

*};*

*return (*

*<CenterPiece>*

*<Card>*

*<CardHeader>{isLogin ? 'Login' : 'Sign Up'}</CardHeader>*

*<CardBody>*

*<ErrorText error={error} />*

*<Button to uzupełniasz na samym końcu nie teraz*

*block*

*disabled={authenticating}*

*onClick={() => SignInWithSocialMedia(Providers.google)}*

*style={{*

*backgroundColor: '#ea4335',*

*borderColor: '#ea4335',*

*}}*

*>*

*<i className="fa-brands fa-google mr-2"></i> Sign{' '}*

*{isLogin ? 'in' : 'up'} with Google*

*</Button>*

*{authenticating && <LoadingComponent card={false} />}        </CardBody>*

*</Card>*

*</CenterPiece>*

*);*

*};*

Potem tworzymy error i sukces komponent żeby wyświetlać komunikaty błędu, sukcesu

*import React from 'react';*

*export interface IErrorTextProps {*

*error: string;*

*}*

*export const ErrorText = (props: IErrorTextProps) => {*

*const { error } = props;*

*if (error) return null;*

*return <small className="text-danger">{error}</small>;*

*};*

*import React from 'react';*

*export interface ISuccessTextProps {*

*success: string;*

*}*

*export const SuccessText = (props: ISuccessTextProps) => {*

*const { success } = props;*

*if (success) return null;*

*return <small className="text-success">{success}</small>;*

*};*

Uzupełniasz login.tsx

*<ErrorText error={error} />*

*<Button*

*block*

*disabled={authenticating}*

*onClick={() => SignInWithSocialMedia(Providers.google)}*

*style={{*

*backgroundColor: '#ea4335',*

*borderColor: '#ea4335',*

*}}*

*>*

*<i className="fa-brands fa-google mr-2"></i> Sign{' '}*

*{isLogin ? 'in' : 'up'} with Google*

*</Button>*

*{authenticating && <LoadingComponent card={false} />}*

Kopiujesz sobie ikone google, z font awesome i dodajesz

Auth po stronie serwera

Tworzymy katalog intrfaces i w nim plik user z interfajsem

*import { Document } from 'mongoose';*

*export default interface IUser extends Document {*

*uid: string;*

*name: string;*

*}*

Dalej tworzymy katalog models i plik user.model z prostym modelem użytkownika

*import IUser from '../interfaces/user.interface';*

*const UserSchema: Schema = new Schema({*

*uid: { type: String, unique: true },*

*name: { type: String }*

*});*

*export default mongoose.model<IUser>('User', UserSchema);*

Dalej tworzymy katalog middleware i plik extractFirebaseInfo.ts

Który pobiera firbase token i sprawdza z firebase admin czy user jest valid jak jest to zwróci dane o userze któRE POTRZEbujemy, a jak nie to wyświetli komunikat ‘not authorized’ i blad 404 JAK PONIŻEJ

*import logging from '../src/config/loggins';*

*import firebaseAdmin from 'firebase-admin';*

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

*const extractFirebaseInfo = (req: Request, res: Response, next: NextFunction) => {*

*logging.info('Validating firebase token, please wait...');*

*/\*\* Bearer abc123.... \*/*

*let token = req.headers.authorization?.split(' ')[1];*

*if (token) {*

*firebaseAdmin*

*.auth()*

*.verifyIdToken(token)*

*.then((result) => {*

*if (result) {*

*/\*\* add info to response \*/*

*res.locals.firebase = result;*

*res.locals.fire\_token = token;*

*next();*

*} else {*

*logging.warn('Token invalid, unauthorized...');*

*return res.status(401).json({*

*message: 'unauthorized'*

*});*

*}*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(401).json({*

*error,*

*message: 'unauthorized'*

*});*

*});*

*} else {*

*return res.status(401).json({*

*message: 'unauthorized'*

*});*

*}*

*};*

*export default extractFirebaseInfo;*

Potem tworzymy katalog controllers i plik user kontroler, który będzie walidował, tworzyl, logowal, odczytywal dane:

Funkcja walidujaca

*import logging from '../config/loggins';*

*import User from '../models/user.model';*

*const validate = (req: Request, res: Response, next: NextFunction) => {*

*logging.info('Token validated, returning user...');*

*let firebase = res.locals.firebase;*

*return User.findOne({ uid: firebase.uid })*

*.then((user) => {*

*if (user) {*

*return res.status(200).json({ user });*

*} else {*

*return res.status(401).json({*

*message: 'user not found'*

*});*

*}*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(500).json({*

*error*

*});*

*});*

Pozostałe funkcje

*const create = async (req: Request, res: Response, next: NextFunction) => {*

*logging.info('attempting to register user ...');*

*let { uid, name } = req.body;*

*let fire\_token = res.locals.fire\_token;*

*const user = new User({*

*\_id: new mongoose.Types.ObjectId(),*

*uid,*

*name    });*

*return await user*

*.save()*

*.then((newUser) => {*

*logging.info(`New user ${uid} created...`);*

*return res.status(201).json({ usser: newUser, fire\_token });*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(500).json({*

*error*

*});*

*});*

*};*

*const login = async (req: Request, res: Response, next: NextFunction) => {*

*logging.info('Logging in ... ...');*

*let { uid } = req.body;*

*let fire\_token = res.locals.fire\_token;*

*return await User.findOne({ uid })*

*.then((user) => {*

*if (user) {*

*logging.info(`User ${uid} found, signing in ...`);*

*return res.status(200).json({ user, fire\_token });*

*} else {*

*logging.info(`User ${uid} not found, register...`);*

*return create(req, res, next);*

*}*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(500).json({*

*error*

*});*

*});*

*};*

*const read = (req: Request, res: Response, next: NextFunction) => {*

*const \_id = req.params.userID;*

*logging.info(`Incoming read for ${\_id} ...`);*

*return User.findById(\_id)*

*.then((user) => {*

*if (user) {*

*return res.status(200).json({ user });*

*} else {*

*return res.status(404).json({ message: 'Not found' });*

*}*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(500).json({*

*error*

*});*

*});*

*};*

*const readAll = (req: Request, res: Response, next: NextFunction) => {*

*logging.info(`Incoming read all ...`);*

*return User.find()*

*.exec()*

*.then((users) => {*

*return res.status(200).json({*

*count: users.length,*

*users*

*});*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(500).json({*

*error*

*});*

*});*

*};*

*export default { validate, login, create, read, readAll };*

MUSI BYC EXPORT DEFAULT BO NIE STORZYSZ SOBIE W RUTERZE ZMIENNEJ DO KTÓRYCH BEDZIESZ PRZYPISYWAĆ SCIEŻKi

KOlejno tworzysz nowy katalog routes i plik user.route.ts

*import controller from '../controllers/user.controller';*

*import extractFirebaseInfo from '../../middleware/extractFirebaseInfo';*

*const router = express.Router();*

*router.get('/validate', extractFirebaseInfo, controller.validate);*

*router.get('/:userID', controller.read);*

*router.post('/create', controller.create);*

*router.post('/login', controller.login);*

*router.get('/', controller.readAll);*

*export = router;*

Idziemy do serwera i importujemy ruter

*/\*\* Routes \*/*

*router.use('/users', userRoutes);*

I dodajemy funkcje w auth.ts w katalogu modules

*import firebase from 'firebase/compat';*

*import { auth } from '../config/firebase';*

*import IUser from '../interfaces/user.interface';*

*import axios from 'axios';*

*import config from '../config/config';*

*import logging from '../config/logging';*

*export const SignInWithSocialMedia = (provider: firebase.auth.AuthProvider) =>*

*new Promise<firebase.auth.UserCredential>((resolve, reject) => {*

*auth*

*.signInWithPopup(provider)*

*.then((result) => resolve(result))*

*.catch((error) => reject(error));*

*});*

*export const Authenticate = async (*

*uid: string,*

*name: string,*

*fire\_token: string,*

*callback: (error: string | null, user: IUser | null) => void*

*) => {*

*try {*

*const response = await axios({*

*method: 'POST',*

*url: `${config.server.url}/users/login`,*

*data: {*

*uid,*

*name,*

*},*

*headers: { Authorization: `Bearer ${fire\_token}` },*

*});*

*if (*

*response.status === 200 ||*

*response.status === 201 ||*

*response.status === 304*

*) {*

*logging.info('authenticated successfully');*

*callback(null, response.data.user);*

*} else {*

*logging.warn('Unable to authenticate.');*

*callback('Unable to authenticate', null);*

*}*

*} catch (error) {*

*logging.error(error);*

*callback('Unable to authenticate.', null);*

*}*

*};*

*export const Validate = async (*

*fire\_token: string,*

*callback: (error: string | null, user: IUser | null) => void*

*) => {*

*try {*

*const response = await axios({*

*method: 'GET',*

*url: `${config.server.url}/users/validate`,*

*headers: { Authorization: `Bearer ${fire\_token}` },*

*});*

*if (response.status === 200 || response.status === 304) {*

*logging.info('validated successfully');*

*callback(null, response.data.user);*

*} else {*

*logging.warn('Unable to validate.');*

*callback('Unable to validate', null);*

*}*

*} catch (error) {*

*logging.error(error);*

*callback('Unable to validate.', null);*

*}*

*};*

Zmiany w pliku application.tsx

*if (fire\_token === null) {*

*userDispatch({ type: 'logout', payload: initialUserState });*

*setAuthState('No credentials found.');*

*setTimeout(() => {*

*setLoading(false);*

*}, 1000);*

*} else {*

*/\*\* validate with BE \*/*

*return Validate(fire\_token, (error, user) => {*

*if (error) {*

*logging.error(error);*

*setAuthState('User not valid, logging out...');*

*userDispatch({ type: 'logout', payload: initialUserState });*

*setTimeout(() => {*

*setLoading(false);*

*}, 1000);*

*} else if (user) {*

*setAuthState('User authenticated...');*

*userDispatch({ type: 'login', payload: { user, fire\_token } });*

*setTimeout(() => {*

*setLoading(false);*

*}, 1000);*

*}*

*});*

*}*

*};*

Wracamy do login page

*import React, { useContext, useState } from 'react';*

*import { useHistory } from 'react-router-dom';*

*import UserContext from '../context/user.context';*

*import IPage from '../interfaces/page.interface';*

*import firebase from 'firebase/compat';*

*import {*

*Authenticate,*

*SignInWithSocialMedia as SocialMediaPopup,*

*} from '../modules/auth';*

*import logging from '../config/logging';*

*import { CenterPiece } from '../components/CenterPiece/CenterPiece';*

*import { Button, Card, CardBody, CardHeader } from 'reactstrap';*

*import { ErrorText } from '../components/ErrorText/ErrorText';*

*import { Providers } from '../config/firebase';*

*import LoadingComponent from '../components/LoadingComponent/Loading';*

*export const LoginPage = (props: IPage) => {*

*const [authenticating, setAuthenticating] = useState<boolean>(false);*

*const [error, setError] = useState<string>('');*

*const userContext = useContext(UserContext);*

*const history = useHistory();*

*const isLogin = window.location.pathname.includes('login');*

*const SignInWithSocialMedia = (provider: firebase.auth.AuthProvider) => {*

*if (error !== '') setError('');*

*setAuthenticating(true);*

*SocialMediaPopup(provider)*

*.then(async (result) => {*

*logging.info(result);*

*let user = result.user;*

*if (user) {*

*let uid = user.uid;*

*let name = user.displayName;*

*if (name) {*

*try {*

*let fire\_token = await user.getIdToken();*

*/\*\* if we get token, auth with BE \*/*

*Authenticate(uid, name, fire\_token, (error, \_user) => {*

*if (error) {*

*setError(error);*

*setAuthenticating(false);*

*} else if (\_user) {*

*userContext.userDispatch({*

*type: 'login',*

*payload: { user: \_user, fire\_token },*

*});*

*history.push('/');*

*}*

*});*

*} catch (error) {*

*setError('Invalid token');*

*logging.error(error);*

*setAuthenticating(false);*

*}*

*} else {*

*/\*\* if no name returned, we could have a custom form here getting users name*

*\* depending on the porvider you are using. google generally return*

*\* ones, lets just use that for now*

*\*/*

*setError('The identity provider doesnt have username');*

*setAuthenticating(false);*

*}*

*} else {*

*setError('Identity provider is missing necessary infrmation');*

*setAuthenticating(false);*

*}*

*})*

*.catch((error) => {*

*setError(error.message);*

*setAuthenticating(false);*

*});*

*};*

*return (*

*<CenterPiece>*

*<Card>*

*<CardHeader>{isLogin ? 'Login' : 'Sign Up'}</CardHeader>*

*<CardBody>*

*<ErrorText error={error} />*

*<Button*

*block*

*disabled={authenticating}*

*onClick={() => SignInWithSocialMedia(Providers.google)}*

*style={{*

*backgroundColor: '#ea4335',*

*borderColor: '#ea4335',*

*}}*

*>*

*<i className="fa-brands fa-google mr-2"></i> Sign{' '}*

*{isLogin ? 'in' : 'up'} with Google*

*</Button>*

*{authenticating && <LoadingComponent card={false} />}*

*</CardBody>*

*</Card>*

*</CenterPiece>*

*);*

*};*

Uzupelniasz navbara o przyciski

*export const Navigation = (props: INavbarProps) => {*

*const userContext = useContext(UserContext);*

*const { user } = userContext.userState;*

*const Logout = () => {*

*userContext.userDispatch({ type: 'logout', payload: initialUserState });*

*};*

*return (*

*<Navbar color="light" light sticky="top" expoand="md">*

*<Container>*

*<NavbarBrand tag={Link} to="/">*

*</NavbarBrand>*

*<Nav className="mr-auto" navbar />*

*{user.\_id === '' ? (*

*<div>*

*<NavbarText tag={Link} to="/login">*

*Login*

*</NavbarText>*

*<NavbarText className="mr-2 ml-2">|</NavbarText>*

*<NavbarText tag={Link} to="/register">*

*Sign in*

*</NavbarText>*

*</div>*

*) : (*

*<div>*

*<Button outline tag={Link} to="/edit">*

*Post a blog*

*</Button>*

*<NavbarText className="mr-2 ml-2">|</NavbarText>*

*<Button outline size="sm" onClick={() => Logout()}>*

*Logout*

*</Button>*

*</div>*

*)}*

*</Container>*

*</Navbar>*

*);*

*};*

Po tym wszystkim tworzymy blog controller. Na backendzie tworzymy interfejs blog.interfce

*import IUser from './user.interface';*

*export default interface IBlog extends Document {*

*title: string;*

*author: IUser; autor to IUser czyli interfejs usera*

*content: string;*

*headline: string;*

*picture?: string;*

*}*

I model blog.model

*const BlogSchema: Schema = new Schema(*

*{*

*title: { type: String, unique: true },*

*author: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },*

*content: { type: String },*

*headline: { type: String },*

*picture: { type: String }*

*},*

*{*

*timestamps: true*

*} autor to w modelu: typ objekt id odnoszący sie do ‘User’ bo tak nazwaliśmy to w user.model.ts export default mongoose.model<IUser>('User', UserSchema);*

*);*

*export default mongoose.model<IBlog>('Blog', BlogSchema);*

Jak to mamy to tworzymy blog kontroler.

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

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

*import express from 'express';*

*import logging from '../config/loggins';*

*import Blog from '../models/blog.model';*

*import mongoose from 'mongoose';*

*const create = (req: Request, res: Response, next: NextFunction) => {*

*logging.info('attempting to register user ...');*

*let { author, title, content, headline, picture } = req.body;*

*const blog = new Blog({*

*\_id: new mongoose.Types.ObjectId(),*

*author,*

*title,*

*content,*

*headline,*

*picture*

*});*

*return blog*

*.save()*

*.then((newBlog) => {*

*logging.info(`New blog created...`);*

*return res.status(201).json({ blog: newBlog });*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(500).json({*

*error*

*});*

*});*

*};*

*const read = (req: Request, res: Response, next: NextFunction) => {*

*const \_id = req.params.blogID;*

*logging.info(`Incoming read for ${\_id} ...`);*

*return Blog.findById(\_id)*

*.then((blog) => {*

*if (blog) {*

*return res.status(200).json({ blog });*

*} else {*

*return res.status(404).json({ message: 'Not found' });*

*}*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(500).json({*

*error*

*});*

*});*

*};*

*const readAll = (req: Request, res: Response, next: NextFunction) => {*

*logging.info(`Incoming read all ...`);*

*Blog.find()*

*.exec()*

*.then((blogs) => {*

*return res.status(200).json({*

*count: blogs.length,*

*blogs: blogs*

*});*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(500).json({*

*error*

*});*

*});*

*};*

*const query = (req: Request, res: Response, next: NextFunction) => {*

*logging.info(`Incoming query ...`);*

*Blog.find(req.body)*

*.exec()*

*.then((blogs) => {*

*return res.status(200).json({*

*count: blogs.length,*

*blogs: blogs*

*});*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(500).json({*

*error*

*});*

*});*

*};*

*const update = (req: Request, res: Response, next: NextFunction) => {*

*const \_id = req.params.blogID;*

*logging.info(`Incoming update for ${\_id} ...`);*

*Blog.findById(\_id)*

*.exec()*

*.then((blog) => {*

*if (blog) {*

*blog.set(req.body);*

*blog.save()*

*.then((newBlog) => {*

*logging.info(`Blog updated...`);*

*return res.status(201).json({ blog: newBlog });*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(500).json({*

*error*

*});*

*});*

*} else {*

*return res.status(404).json({ message: 'Not found' });*

*}*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(500).json({*

*error*

*});*

*});*

*};*

*const deleteBlog = async (req: Request, res: Response, next: NextFunction) => {*

*const \_id = req.params.blogID;*

*logging.warn(`Incoming delete for ${\_id} ...`);*

*return await Blog.findByIdAndDelete(\_id)*

*.then(() => {*

*return res.status(200).json({ message: 'Blog deleted' });*

*})*

*.catch((error) => {*

*logging.error(error);*

*return res.status(500).json({*

*error*

*});*

*});*

*};*

*export default { create, read, readAll, update, query, deleteBlog };*

I blog.routes.ts

*import express from 'express';*

*import controller from '../controllers/blog.controller';*

*const router = express.Router();*

*router.get('/:userID', controller.read);*

*router.post('/create', controller.create);*

*router.post('/query', controller.query);router.post('/update/:blogID', controller.update);*

*router.delete('/:blogID', controller.deleteBlog);*

*router.get('/', controller.readAll);*

*export = router;*

Dodajesz routing do server.ts

*import blogRoutes from './routes/blog.route';*

...............

*router.use('/blogs', blogRoutes);*

Idziemy na FE tworzymy interfejs dla bloga rozszerzony o dane z mongo:

*import IUser from './user.interface';*

*export default interface IBlog {*

*\_id: string;*

*title: string;*

*author: string | IUser;*

*content: string;*

*headline: string;*

*picture?: string;*

*createdAt: string;*

*updatedAt: string;*

*}*

I dalej na home.pge

*import axios from 'axios';*

*import React, { useEffect, useState } from 'react';*

*import { Link } from 'react-router-dom';*

*import { Container } from 'reactstrap';*

*import { BlogPreview } from '../components/BlogPreview/BlogPreview';*

*import { Header } from '../components/header/Header';*

*import LoadingComponent from '../components/LoadingComponent/Loading';*

*import { Navigation } from '../components/navigation/Navbar';*

*import config from '../config/config';*

*import logging from '../config/logging';*

*import IBlog from '../interfaces/blog.interface';*

*import IUser from '../interfaces/user.interface';*

*export const HomePage = () => {*

*const [blogs, setBlogs] = useState<IBlog[]>([]);*

*const [loading, setLoading] = useState<boolean>(true);*

*const [error, setError] = useState<string>('');*

*useEffect(() => {*

*GetAllBlogs();*

*}, []);*

*const GetAllBlogs = async () => {*

*try {*

*const response = await axios({*

*method: 'GET',*

*url: `${config.server.url}/blogs`,*

*});*

*if (response.status === 200 || response.status === 304) {*

*let blogs = response.data.blogs as IBlog[];*

*/\*\* newest blogs comes first -compare function \*/*

*blogs.sort((x, y) => y.updatedAt.localeCompare(x.updatedAt));*

*setBlogs(blogs);*

*}*

*} catch (error) {*

*logging.error(error);*

*setError('Unable to retrieve blogs');*

*} finally {*

*setTimeout(() => {*

*setLoading(false);*

*}, 1000);*

*}*

*};*

*if (loading) {*

*return <LoadingComponent>Loading blogs...</LoadingComponent>;*

*}*

*return (*

*<Container fluid className="p-0">*

*<Navigation />*

*<Header title="Nerdy blog side..." headline="you maust check it out!!" />*

*<Container className="mt-5">*

*{blogs.length === 0 && (*

*<p>*

*There are no blogs yet, you should <Link to="/edit">post</Link>one*

*</p>*

*)}*

*{blogs.map((blog, index) => {*

*return (*

*<div key={index}>*

*<BlogPreview*

*\_id={blog.\_id}*

*author={(blog.author as IUser).name}*

*headline={blog.headline}*

*title={blog.title}*

*createdAt={blog.createdAt}*

*updatedAt={blog.updatedAt}*

*/>*

*<hr />*

*</div>*

*);*

*})}*

*</Container>*

*</Container>*

*);*

*};*

Potem blog.preview

*import React from 'react';*

*import { Link } from 'react-router-dom';*

*import { Card, CardBody } from 'reactstrap';*

*export interface IBlogPreviewProps {*

*\_id: string;*

*title: string;*

*headline: string;*

*author: string;*

*createdAt: string;*

*updatedAt: string;*

*children?: any;*

*}*

*export const BlogPreview = (props: IBlogPreviewProps) => {*

*const { \_id, author, children, createdAt, updatedAt, headline, title } =*

*props;*

*return (*

*<Card className="border-0">*

*<CardBody className="p-0">*

*<Link*

*to={`/blogs/${\_id}`}*

*style={{ textDecoration: 'none' }}*

*className="text-primary"*

*>*

*<h1>*

*<strong>{title}</strong>*

*</h1>*

*<h3>{headline}</h3>*

*<br />*

*</Link>*

*{createdAt !== updatedAt ? (*

*<p>*

*Updated by {author} at {new Date(updatedAt).toLocaleString()}*

*</p>*

*) : (*

*<p>*

*Posted by {author} at {new Date(createdAt).toLocaleString()}*

*</p>*

*)}*

*{children}*

*</CardBody>*

*</Card>*

*);*

*};*

Na be w blog controler wrzucasz

*.populate('author')*

W funkcjach query, read, readALL

Instalujesz paczki react-draft-wysiwyg draftjs-to-html html-to-draftjs

I typy do ts @types/react-draft-wysiwyg @types/draftjs-to-html @types/html-to-draftjs. POTEM IDZIEMY DO STRONY EDYCJI

*import React, { useContext, useEffect, useState } from 'react';*

*import IPage from '../interfaces/page.interface';*

*import { EditorState, ContentState, convertToRaw } from 'draft-js';*

*import UserContext from '../context/user.context';*

*import { Link, RouteComponentProps, withRouter } from 'react-router-dom';*

*import config from '../config/config';*

*import axios from 'axios';*

*import logging from '../config/logging';*

*import IBlog from '../interfaces/blog.interface';*

*import htmlToDraft from 'html-to-draftjs';*

*import draftToHtml from 'draftjs-to-html';*

*import LoadingComponent from '../components/LoadingComponent/Loading';*

*import { Button, Container, Form, FormGroup, Input, Label } from 'reactstrap';*

*import { Navigation } from '../components/navigation/Navbar';*

*import { Header } from '../components/header/Header';*

*import { ErrorText } from '../components/ErrorText/ErrorText';*

*import { Editor } from 'react-draft-wysiwyg';*

*import { SuccessText } from '../components/SuccessText/SuccessText';*

*import 'react-draft-wysiwyg/dist/react-draft-wysiwyg.css';*

*export const EditPage = (props: IPage & RouteComponentProps<any>) => {*

*const [\_id, setId] = useState<string>('');*

*const [title, setTitle] = useState<string>('');*

*const [picture, setPicture] = useState<string>('');*

*const [content, setContent] = useState<string>('');*

*const [headline, setHeadline] = useState<string>('');*

*const [editorState, setEditorState] = useState<EditorState>(*

*EditorState.createEmpty()*

*);*

*const [saving, setSaving] = useState<boolean>(false);*

*const [loading, setLoading] = useState<boolean>(true);*

*const [success, setSuccess] = useState<string>('');*

*const [error, setError] = useState<string>('');*

*const { user } = useContext(UserContext).userState;*

*useEffect(() => {*

*let blogID = props.match.params.blogID;*

*if (blogID) {*

*setId(blogID);*

*getBlog(blogID);*

*} else {*

*setLoading(false);*

*}*

*}, []);*

*const getBlog = async (id: string) => {*

*try {*

*const response = await axios({*

*method: 'GET',*

*url: `${config.server.url}/blogs/read/${id}`,*

*});*

*if (response.status === 200 || response.status === 304) {*

*if (user.\_id !== response.data.blog.author.\_id) {*

*logging.warn('This blog is owned by someone else!');*

*setId('');*

*} else {*

*let blog = response.data.blog as IBlog;*

*setTitle(blog.title);*

*setContent(blog.content);*

*setHeadline(blog.headline);*

*setPicture(blog.picture || '');*

*/\*\* convert html string to draft JS editor state \*/*

*const contentBlock = htmlToDraft(blog.content);*

*const contentState = ContentState.createFromBlockArray(*

*contentBlock.contentBlocks*

*);*

*const \_editorState = EditorState.createWithContent(contentState);*

*setEditorState(\_editorState);*

*}*

*} else {*

*setError(`Unable to retriev blog ${\_id}`);*

*setId('');*

*}*

*} catch (error: any) {*

*setError(error.message);*

*} finally {*

*setLoading(false);*

*}*

*};*

*const createBlog = async () => {*

*if (title === '' || headline === '' || content === '') {*

*setError('Please fill out all required fields...');*

*setSuccess('');*

*return null;*

*}*

*setError('');*

*setSuccess('');*

*setSaving(true);*

*try {*

*const response = await axios({*

*method: 'POST',*

*url: `${config.server.url}/blogs/create`,*

*data: {*

*title,*

*picture,*

*headline,*

*content,*

*author: user.\_id,*

*},*

*});*

*if (response.status === 201) {*

*setId(response.data.blog.\_id);*

*setSuccess('Blog posted. You can continue to edit on this page.');*

*} else {*

*setError('Unable to save blog.');*

*}*

*} catch (error: any) {*

*setError(error.message);*

*} finally {*

*setSaving(false);*

*}*

*};*

*const editBlog = async () => {*

*if (title === '' || headline === '' || content === '') {*

*setError('Please fill out all required fields...');*

*setSuccess('');*

*return null;*

*}*

*setError('');*

*setSuccess('');*

*setSaving(true);*

*try {*

*const response = await axios({*

*method: 'PATCH',*

*url: `${config.server.url}/blogs/update/${\_id}`,*

*data: {*

*title,*

*picture,*

*headline,*

*content,*

*author: user.\_id,*

*},*

*});*

*if (response.status === 201) {*

*setSuccess('Blog updated.');*

*} else {*

*setError('Unable to save blog.');*

*}*

*} catch (error: any) {*

*setError(error.message);*

*} finally {*

*setSaving(false);*

*}*

*};*

*if (loading) return <LoadingComponent>Loading editor...</LoadingComponent>;*

*return (*

*<Container fluid className="p-0">*

*<Navigation />*

*<Header*

*headline=""*

*title={\_id !== '' ? 'Edit your blog' : 'Create a new blog'}*

*/>*

*<Container className="mt-5 mb-5">*

*<ErrorText error={error} />*

*<Form>*

*<FormGroup>*

*<Label for="title">Title \*</Label>*

*<Input*

*type="text"*

*name="title"*

*value={title}*

*id="title"*

*placeholder="insert a title..."*

*disabled={saving}*

*onChange={(event) => setTitle(event.target.value)}*

*/>*

*</FormGroup>*

*<FormGroup>*

*<Label for="picture">Picture url</Label>*

*<Input*

*type="text"*

*name="picture"*

*value={picture}*

*id="picture"*

*placeholder="picture url, ex: http://..."*

*disabled={saving}*

*onChange={(event) => setPicture(event.target.value)}*

*/>*

*</FormGroup>*

*<FormGroup>*

*<Label for="headline">Headline \*</Label>*

*<Input*

*type="text"*

*name="headline"*

*value={headline}*

*id="headline"*

*placeholder="insert headline..."*

*disabled={saving}*

*onChange={(event) => setHeadline(event.target.value)}*

*/>*

*</FormGroup>*

*<FormGroup>*

*<Label>Content</Label>*

*<Editor*

*editorState={editorState}*

*wrapperClassName="card"*

*editorClassName="card-body"*

*onEditorStateChange={(newState) => {*

*setEditorState(newState);*

*setContent(*

*draftToHtml(convertToRaw(newState.getCurrentContent()))*

*);*

*}}*

*toolbar={{*

*options: [*

*'inline',*

*'blockType',*

*'fontSize',*

*'list',*

*'textAlign',*

*'history',*

*'embedded',*

*'emoji',*

*'image',*

*],*

*inline: { inDropdown: true },*

*list: { inDropdown: true },*

*textAlign: { inDropdown: true },*

*link: { inDropdown: true },*

*history: { inDropdown: true },*

*}}*

*/>*

*</FormGroup>*

*<FormGroup>*

*<SuccessText success={success} />*

*</FormGroup>*

*<FormGroup>*

*<Button*

*block*

*onClick={() => {*

*if (\_id !== '') {*

*editBlog();*

*} else {*

*createBlog();*

*}*

*}}*

*disabled={saving}*

*>*

*<i className="fas fa-save mr-1"></i>*

*{\_id !== '' ? 'Update' : 'Post'}*

*</Button>*

*{\_id !== '' && (*

*<Button block color="success" tag={Link} to={`/blogs/${\_id}`}>*

*View your post!*

*</Button>*

*)}*

*</FormGroup>*

*<FormGroup>*

*<Label>Preview</Label>*

*<div className="border p-2">*

*<div*

*dangerouslySetInnerHTML={{*

*\_\_html: content,*

*}}*

*></div>*

*</div>*

*</FormGroup>*

*</Form>*

*</Container>*

*</Container>*

*);*

*};*

*export default withRouter(EditPage);*

Dodajesz male z miany w ruterze usera

*router.get('/read/:userID', controller.read);*

I blog ruter

*router.get('/read/:blogID', controller.read);*

Tworzymy strone bloga blog page

*import React, { useContext, useEffect, useState } from 'react';*

*import IPage from '../interfaces/page.interface';*

*import { EditorState, ContentState, convertToRaw } from 'draft-js';*

*import UserContext from '../context/user.context';*

*import { Link, RouteComponentProps, withRouter } from 'react-router-dom';*

*import config from '../config/config';*

*import axios from 'axios';*

*import logging from '../config/logging';*

*import IBlog from '../interfaces/blog.interface';*

*import htmlToDraft from 'html-to-draftjs';*

*import draftToHtml from 'draftjs-to-html';*

*import LoadingComponent from '../components/LoadingComponent/Loading';*

*import { Button, Container, Form, FormGroup, Input, Label } from 'reactstrap';*

*import { Navigation } from '../components/navigation/Navbar';*

*import { Header } from '../components/header/Header';*

*import { ErrorText } from '../components/ErrorText/ErrorText';*

*import { Editor } from 'react-draft-wysiwyg';*

*import { SuccessText } from '../components/SuccessText/SuccessText';*

*import 'react-draft-wysiwyg/dist/react-draft-wysiwyg.css';*

*export const EditPage = (props: IPage & RouteComponentProps<any>) => {*

*const [\_id, setId] = useState<string>('');*

*const [title, setTitle] = useState<string>('');*

*const [picture, setPicture] = useState<string>('');*

*const [content, setContent] = useState<string>('');*

*const [headline, setHeadline] = useState<string>('');*

*const [editorState, setEditorState] = useState<EditorState>(*

*EditorState.createEmpty()*

*);*

*const [saving, setSaving] = useState<boolean>(false);*

*const [loading, setLoading] = useState<boolean>(true);*

*const [success, setSuccess] = useState<string>('');*

*const [error, setError] = useState<string>('');*

*const { user } = useContext(UserContext).userState;*

*useEffect(() => {*

*let blogID = props.match.params.blogID;*

*if (blogID) {*

*setId(blogID);*

*getBlog(blogID);*

*} else {*

*setLoading(false);*

*}*

*}, []);*

*const getBlog = async (id: string) => {*

*try {*

*const response = await axios({*

*method: 'GET',*

*url: `${config.server.url}/blogs/read/${id}`,*

*});*

*if (response.status === 200 || response.status === 304) {*

*if (user.\_id !== response.data.blog.author.\_id) {*

*logging.warn('This blog is owned by someone else!');*

*setId('');*

*} else {*

*let blog = response.data.blog as IBlog;*

*setTitle(blog.title);*

*setContent(blog.content);*

*setHeadline(blog.headline);*

*setPicture(blog.picture || '');*

*/\*\* convert html string to draft JS editor state \*/*

*const contentBlock = htmlToDraft(blog.content);*

*const contentState = ContentState.createFromBlockArray(*

*contentBlock.contentBlocks*

*);*

*const \_editorState = EditorState.createWithContent(contentState);*

*setEditorState(\_editorState);*

*}*

*} else {*

*setError(`Unable to retriev blog ${\_id}`);*

*setId('');*

*}*

*} catch (error: any) {*

*setError(error.message);*

*} finally {*

*setLoading(false);*

*}*

*};*

*const createBlog = async () => {*

*if (title === '' || headline === '' || content === '') {*

*setError('Please fill out all required fields...');*

*setSuccess('');*

*return null;*

*}*

*setError('');*

*setSuccess('');*

*setSaving(true);*

*try {*

*const response = await axios({*

*method: 'POST',*

*url: `${config.server.url}/blogs/create`,*

*data: {*

*title,*

*picture,*

*headline,*

*content,*

*author: user.\_id,*

*},*

*});*

*if (response.status === 201) {*

*setId(response.data.blog.\_id);*

*setSuccess('Blog posted. You can continue to edit on this page.');*

*} else {*

*setError('Unable to save blog.');*

*}*

*} catch (error: any) {*

*setError(error.message);*

*} finally {*

*setSaving(false);*

*}*

*};*

*const editBlog = async () => {*

*if (title === '' || headline === '' || content === '') {*

*setError('Please fill out all required fields...');*

*setSuccess('');*

*return null;*

*}*

*setError('');*

*setSuccess('');*

*setSaving(true);*

*try {*

*const response = await axios({*

*method: 'PATCH',*

*url: `${config.server.url}/blogs/update/${\_id}`,*

*data: {*

*title,*

*picture,*

*headline,*

*content,*

*//   author: user.\_id,*

*},*

*});*

*if (response.status === 201) {*

*setSuccess('Blog updated.');*

*} else {*

*setError('Unable to save blog.');*

*}*

*} catch (error: any) {*

*setError(error.message);*

*} finally {*

*setSaving(false);*

*}*

*};*

*if (loading) return <LoadingComponent>Loading editor...</LoadingComponent>;*

*return (*

*<Container fluid className="p-0">*

*<Navigation />*

*<Header*

*image="https://startbootstrap.github.io/startbootstrap-clean-blog/img/home-bg.jpg"*

*headline=""*

*title={\_id !== '' ? 'Edit your blog' : 'Create a new blog'}*

*/>*

*<Container className="mt-5 mb-5">*

*<ErrorText error={error} />*

*<Form>*

*<FormGroup>*

*<Label for="title">Title \*</Label>*

*<Input*

*type="text"*

*name="title"*

*value={title}*

*id="title"*

*placeholder="insert a title..."*

*disabled={saving}*

*onChange={(event) => {*

*setTitle(event.target.value);*

*}}*

*/>*

*</FormGroup>*

*<FormGroup>*

*<Label for="picture">Picture url</Label>*

*<Input*

*type="text"*

*name="picture"*

*value={picture}*

*id="picture"*

*placeholder="picture url, ex: http://..."*

*disabled={saving}*

*onChange={(event) => {*

*setPicture(event.target.value);*

*}}*

*/>*

*</FormGroup>*

*<FormGroup>*

*<Label for="headline">Headline \*</Label>*

*<Input*

*type="text"*

*name="headline"*

*value={headline}*

*id="headline"*

*placeholder="insert headline..."*

*disabled={saving}*

*onChange={(event) => {*

*setHeadline(event.target.value);*

*}}*

*/>*

*</FormGroup>*

*<FormGroup>*

*<Label>Content</Label>*

*<Editor*

*editorState={editorState}*

*wrapperClassName="card"*

*editorClassName="card-body"*

*onEditorStateChange={(newState) => {*

*setEditorState(newState);*

*setContent(*

*draftToHtml(convertToRaw(newState.getCurrentContent()))*

*);*

*}}*

*toolbar={{*

*options: [*

*'inline',*

*'blockType',*

*'fontSize',*

*'list',*

*'textAlign',*

*'history',*

*'embedded',*

*'emoji',*

*'image',*

*],*

*inline: { inDropdown: true },*

*list: { inDropdown: true },*

*textAlign: { inDropdown: true },*

*link: { inDropdown: true },*

*history: { inDropdown: true },*

*}}*

*/>*

*</FormGroup>*

*<FormGroup>*

*<SuccessText success={success} />*

*</FormGroup>*

*<FormGroup>*

*<Button*

*block*

*onClick={() => {*

*if (\_id !== '') {*

*editBlog();*

*} else {*

*createBlog();*

*}*

*}}*

*disabled={saving}*

*>*

*<i className="fas fa-save mr-1"></i>*

*{\_id !== '' ? 'Update' : 'Post'}*

*</Button>*

*{\_id !== '' && (*

*<Button block color="success" tag={Link} to={`/blogs/${\_id}`}>*

*View your post!*

*</Button>*

*)}*

*</FormGroup>*

*<FormGroup>*

*<Label>Preview</Label>*

*<div className="border q1-container p-2">*

*<div*

*dangerouslySetInnerHTML={{*

*\_\_html: content,*

*}}*

*></div>*

*</div>*

*</FormGroup>*

*</Form>*

*</Container>*

*</Container>*

*);*

*};*

*export default withRouter(EditPage);*