POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA
SPECJALNOŚĆ: INŻYNIERIA SYSTEMÓW INFORMATYCZNYCH

PROJEKT ZESPOŁOWY

Masters of Scruum

AUTORZY:

Michał Juszczyk, Łukasz Handschuh, Maciek Jaroński,
Stepanenko Nikita, Vladyslav Lutsenko

NADZORUJĄCY PRACĘ:

Dr Paweł Rogaliński, Jan Niekowal

OCENA PRACY:

WROCŁAW, 2020

Spis treści

Spis rysunków

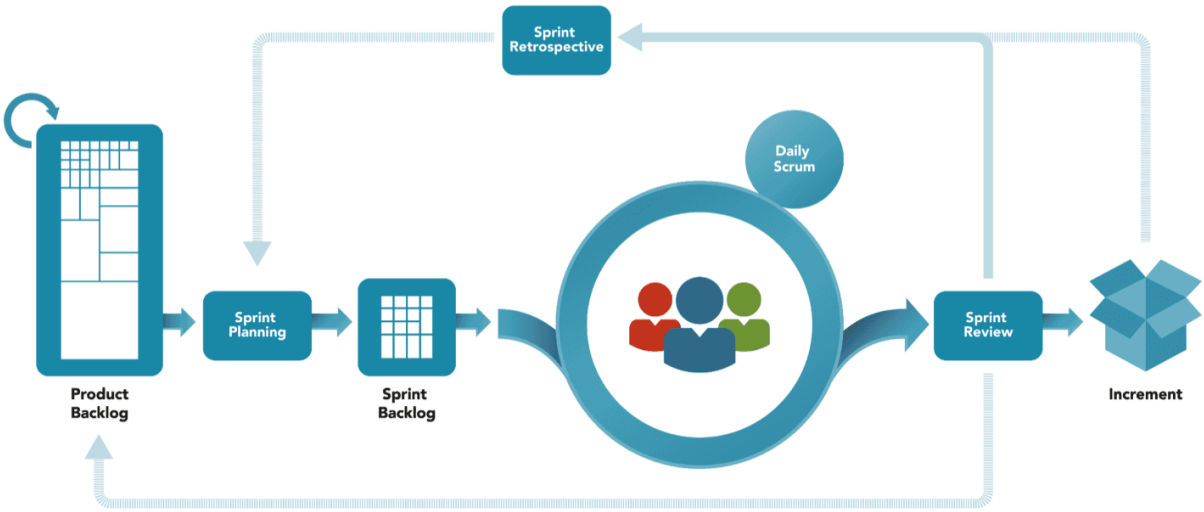
List of Listings

Rozdział 1

Scrum

1.1. Czym jest scrum

Scrum to po prostu zwinne podejście do tworzenia różnego rodzaju produktów - głównie oprogramowania. Polega na podzieleniu produkcji na sprinty, w których każdy doda jakąś funkcjonalność lub poprawi już istniejące rozwiązanie. Sprawia to, że gotowy produkt powstaje wcześnie i jest następnie rozwijany. Umożwia to łatwe śledzenie rozwoju projektu i planowanie kolejnych etapów. W scrumie wyróżnia się trzy role osób: *Product Owner, Scrum Master i Development Team*, a każda z nich ma w projekcie konkretne zadania, które zostały omówione poniżej.



Rys. 1.1: Metodyka Scrum

1.2. Role Scrum

1.2.1. Product Owner

Product Owner to osoba, która decyduje w jakim kierunku ma się rozwijać dany produkt. Ma swoją wizję ostatecznej wersji, a także ustala, które funkcjonalności są ważniejsze i muszą zostać wykonane najpierw. Może też stwierdzić, że dana implementacja rozwiązania nie jest satysfakcjonująca i zlecić poprawki.

Zdaża się, że Product Ownerem nie jest sam właściciel aplikacji lub klient, który ją zlecił, a osoba, która ma kontakt z klientem i zna jego oczekiwania.

1.2.2. Scrum Master

Zadaniem Scrum Mastera jest motywowanie całego zespołu do działania, a także pilnowania, aby wszystkie elementy scruma (m.in. Daily Scrum) były wykonywane przez zespół. Jest to osoba, której głównym zadaniem jest zwiększenie efektywności zespołu poprzez implementację scruma i pilnowanie jego przestrzegania.

1.2.3. Development Team

Cały zespół odpowiedzialny za tworzenie produktu, który otrzymuje konkretne zadania na sprint to Development Team. Zadania mogą być przydzielone dla całego zespołu i stopniowo wykonywane przez członków zespołu lub od razu przyporządkowane do każdego z członków osobno.

1.3. Sprint

Sprint to zaplanowany okres od tygodnia do miesiąca (najczęściej ok. 2 tygodni), w którym zespół dostaje do wykonania określone zadania.

Można wyróżnić kilka etapów sprintu. Na początku wykonuje się planowanie sprintu. Codziennie podczas jego trwania wykonuje się Daily Scrum, a w ustalonym czasie sprintu dodatkowo odbywa się Refinement. Na sam koniec wykonuje się ocenę całego sprintu.

1.3.1. Planowanie

Planowanie to etap, rozpoczynający każdy sprint, wymaga spotkania się PO z zespołem. Product Owner w backlogu wyznacza zadania, które chce wykonać w hierarchii od najważniejszego do najmniej ważnego. Na podstawie tego backlogu ustala się sprint. W zależności od wyceny zadania, dobiera się je, aby ilość zadań jak najlepiej odpowiadała możliwością ich wykonania przez zespół.

1.3.2. Daily Scrum

Daily Scrum to krótkie, codzienne spotkania, które nie powinny trwać więcej niż 30 minut. Na tych spotkaniach, każdy z członków zespołu określa zadania, które będzie wykonywał tego dnia. Dodatkowo podsumowany jest poprzedni dzień - w jakim stopniu udało się wykonać zaplanowane zadanie oraz jakie trudności się napotkało.

1.3.3. Refinement

Spotkanie, które polega na podsumowaniu dotychczas wykonanych działań wykonanych podczas sprintu nosi nazwę Refinement. Podczas tego spotkania można ustalić w jakim stopniu praca wykonana podczas sprintu jest zgodna z planami. W przypadku wszystkich odstępstw można dopasować plan sprintu do zaistniałych sytuacji.

1.3.4. Podsumowanie

Podsumowanie przyjmuje formę spotkania, podczas którego ocenia się pracę wykonaną podczas sprintu i podsumowuje, jak dobrze udało się zrealizować założenia. Często etap ten jest łączony z planowaniem następnego sprintu.

Rozdział 2

Aplikacja

2.1. Założenia

Celem naszej aplikacji jest ułatwienie pracy Scrum Mastera, który odpowiedzialny jest również za to, aby pilnować czy zespół przestrzega ceremonii Scruma, przede wszystkim spotkań takich jak np.: Daily, Retrospective, ale również również innych koniecznych, które nie wynikają bezpośrednio z metodyki Scruma, ale są konieczne do kontynuowania pracy jak np. spotkania między zespołami.

Aplikacja będzie umożliwiała takie rzeczy jak:

- Tworzenie własnego konta.
- Tworzenie oraz edycja grup projektowych.
- Ustawianie czasu, miejsca oraz typu spotkania w obrębie grupy.
- Przypominanie o nadchodzącym spotkaniu.
- Sortowanie spotkań po dacie oraz grupie projektowej.
- Nadawanie roli użytkownikom w obrębie grupy.

2.2. Technologia

2.2.1. Angular

Angular jest frameworkiem *Javascriptu*. Jest to rozbudowane narzędzie umożliwiające tworzenie rozbudowanych aplikacji za pomocą tego właśnie języka.

Angular jest drugim najpopularniejszym rozszerzeniem do JS. Wyprzedza go jedynie React, który jest biblioteką dużo prostszą niż Angular. Aplikacje stworzone przy pomocy Reacta są prostsze i mniejsze, ale Angular pozwala nam na stworzenie całego serwisu i zarządzanie nim w łatwy sposób.

Angular umożliwia wykorzystanie niemal wszystkich narzędzi, jakie zostały stworzone dla Javascriptu. Jeżeli jednak będzie nam brakowało jakiejś funkcjonalności, zawsze możemy pobrać dodatkowe pakiety używając menagera pakietów - npm.

Dodatkowo wykorzystuje Typescript, czyli rozszerzenie języka Javascript. Dzięki niemu mamy większe możliwości panowania nad typani naszych zmiennych i umożliwia lepszą organizację kodu niż zwykły Javascript

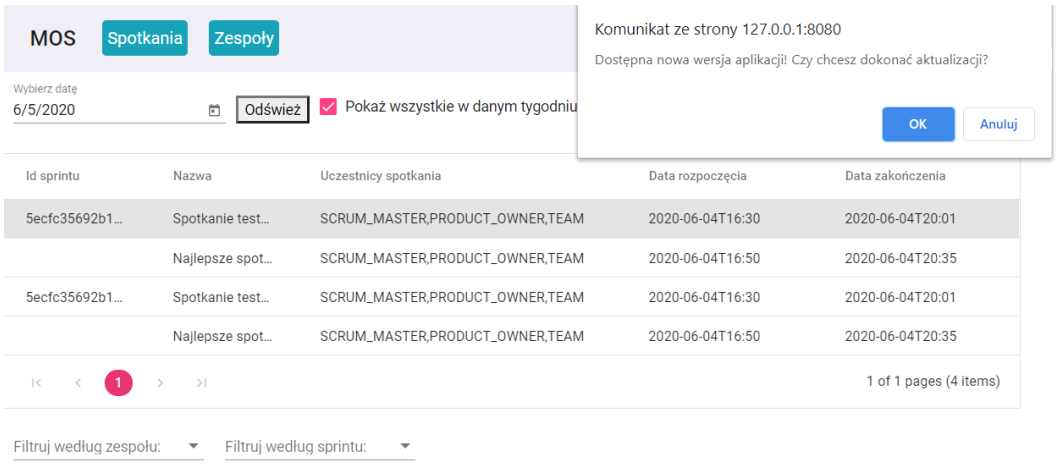
2.2.2. Technologia hybrydowa

Aplikacja została uruchomiona jako aplikacja PWA (ang. Progressive Web App). Jest to bardzo popularny już standard aplikacji. PWA, działa jak aplikacja internetowa, można nią zarządzać z poziomu przeglądarki internetowej lecz użytkownik ma również możliwość zainstalować ją na własnym urządzeniu, zarówno na komputerze jak i urządzeniach mobilnych.

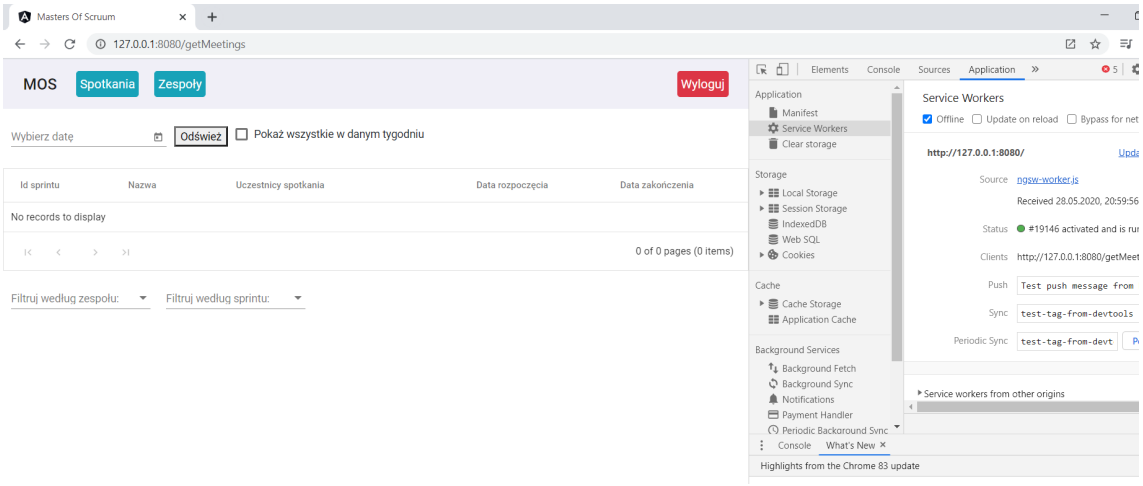
Rys. 2.1: Aplikacja zainstalowana na komputerze

Zastosowanie PWA dało użytkownikowi następujące korzyści:

- Użytkownik, może korzystać z aplikacji w trybie offline. Bazuje ona wtedy na danych które pobrała przy ostatnim uruchomieniu aplikacji w trybie online.

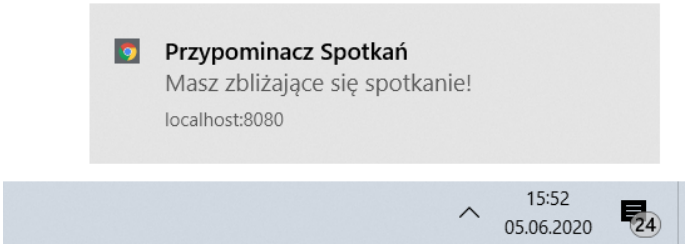


Rys. 2.2: Aktualizacja aplikacji po włączeniu w trybie online



Rys. 2.3: Aplikacja działająca w trybie offline

- • Możliwość uruchomienia aplikacji zarówno na komputerze, jak i urządzeniach mobilnych z system Android oraz iOS.
- • Otrzymywania od aplikacji powiadomień (push-notification) Użytkownik dostaje powiadomienia o zbliżającym się spotkaniu.



Rys. 2.4: Powiadomienie zbliżającego się spotkania

2.2.3. Syncfusion

Do wyświetlania danych w prosty dla użytkownika sposób wykorzystany został pakiet Syncfusion. Jest on darmowy dla celów prywatnych i małych firm. Jest to bardzo potężne narzędzie do zarządzania wyświetlania danymi i edycji danych w prosty dla użytkownika sposób, jednocześnie pozwalający zablokować opcje, do których użytkownik nie powinien mieć dostępu. Pakiet ten można w uproszczeniu nazwać Excelem w przeglądarce.

2.2.4. Back-end

Dla aplikacji Backendowej został użyty framework javy SpringBoot jako podstawa. Jako menedżer pakietów został wybrany maven. Dla autoryzacji wykorzystane są JWT tokeny. W jakości czasowej bazy danych dla starych tokenów została wykorzystana baza danych Redis. Dla połączenia między backendem i bazą danych wykorzystane technologie Hibernate: Repository, Template, Criteria. Narzędzia ułatwiające pracę nad projektem: Lombok (automatyczne ge-

nerowanie getterów i setterów) oraz OpenApi wraz ze swagger-ui (automatyczne generowanie dokumentacji kontrolerów).

2.2.5. Baza danych

Nasza główną bazą danych było MongoDB. Jest to baza nierelacyjna (no SQL), która cechuje się tym, że nie używa ona języka SQL, lecz JavaScript, posiada kolekcję oraz dokumenty zamiast tabeli i wierszy. Każdy dokument jest niezależnym rekordem i może posiadać różne pola. Zaletami tej bazy jest skalowalność (horyzontalna i wertykalna), szybkie działanie nawet dla big data.

W bazie znalazły się takie kolekcje jak:

- użytkownik,
- zespół,
- spotkania
- sprint,
- rola.

2.2.6. Docker

Docker — oprogramowanie do automatyzacji wdrażania i zarządzania aplikacjami w środowiskach obsługujących konteneryzację. Konteneryzacja - metoda wirtualizacji, w której jądro systemu operacyjnego obsługuje wiele izolowanych instancji przestrzeni użytkownika zamiast jednego. Wszystkie aplikację powstają w izolowanych kontenerach. Dla tworzenia obrazu kontenerowego, z którego są uruchomiane kontenery, zawierającego wszystkie niezbędne biblioteki i pliki, wykorzystują się pliki Dockerfile. Dla szybkiego urichomiania kilku aplikacji został wykorzystany docker-compose. Compose używa specjalnego pliku do konfigurowania usług aplikacji(docker-compose.yml). Następnie, za pomocą prostego polecenia utworzą się i uruchomiają się wszystkie usługi z pliku konfiguracyjnego.

2.2.7. Git

Do pracy w zespole wykorzystaliśmy repozytorium Git na platformie GitLab. Dzięki temu możliwa była równoległa praca całego zespołu, co umożliwiło stworzenie tej aplikacji w takim terminie. Repozytorium jest publiczne i dostępne pod adresem https://gitlab.com/mjuszczyk98/masters-of-scruum.

Rozdział 3

Opis architektury

3.1. Frontend

3.1.1. Landing Page

Po załadowaniu aplikacji w przeglądarce naszym oczom ukaże się strona główna. Jest to prosta, minimalistyczna strona, na której przedstawione zostały podstawowe informacje dotyczące aplikacji. Dodatkowo dla poprawy estetyki został dodany obrazek. Strona główna, tak jak cała aplikcja, zostałą wykonana responsywnie, to znaczy, że po zmniejszeniu okna, wszystkie komponenty poukładają się, aby dostosować do ekranu. W przypadku strony głównej mniej znaczący obrazek przesunie się pod tekstem z opsiem aplikacji.

Rys. 3.1: Ekran powitalny

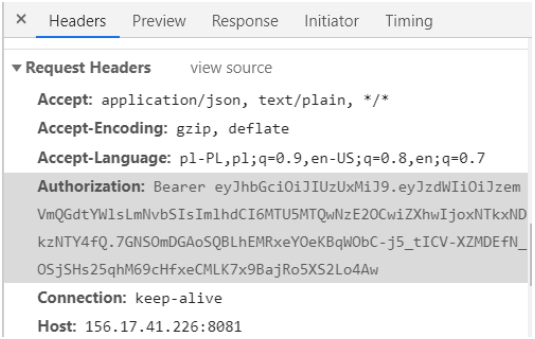
3.1.2. Logowanie

W celu utworzenia autoryzacji użytkownika przy logowaniu, zastosowano standard *JSON Web Token*. Na początku napisano serwis, token-storage.service zarządzający zapisywaniem oraz odczytywaniem tokena w session storage.

```
public saveToken(token: string){
    window.sessionStorage.removeItem(TOKEN_KEY);
    window.sessionStorage.setItem(TOKEN_KEY, token);
}
public getToken(): string{
    return sessionStorage.getItem(TOKEN_KEY);
}
```

Listing 1: Fragment token-storage,service

Zastosowano *Http Interceptors* w celu, zautomatyzowania dołączania do żądań Http informacji wymaganych do autoryzacji. Dokleja on nagłówek **'Authorization'** a w nim Token, z prefiksem 'Bearer'



Rys. 3.2: Nagłówek doklejony przez Http Interceptor

```
import {HTTP_INTERCEPTORS} from '@angular/common/http';
import {Injectable,Injector} from '@angular/core';
import {HttpInterceptor,HttpHandler,HttpRequest} from '@angular/common/http';

import {TokenStorageService} from '../services/token-storage.service';

const TOKEN_HEADER_KEY='Authorization';

@Injectable()
export class AuthInterceptor implements HttpInterceptor{

    constructor (private injector: Injector,private token: TokenStorageService){}
    intercept(req,next){
    let tokenService= this.injector.get(TokenStorageService)
    let authReq=req.clone({
        setHeaders:{
            Authorization:`Bearer ${this.token.getToken()}`
        }
    })
    return next.handle(authReq)
    }
}
export const authInterceptorProviders = [{
    provide: HTTP_INTERCEPTORS,uceClass: AuthInterceptor,multi: true
}];
```

Listing 2: Plik auth.interceptor.ts

Logowanie odbywa się za pomocą komponentu 'login.component'. Za pomocą serwisu do logowania wykorzystuje on obiekty Observable zawierające danego użytkownika wg wpisanego e-mail oraz hasła.

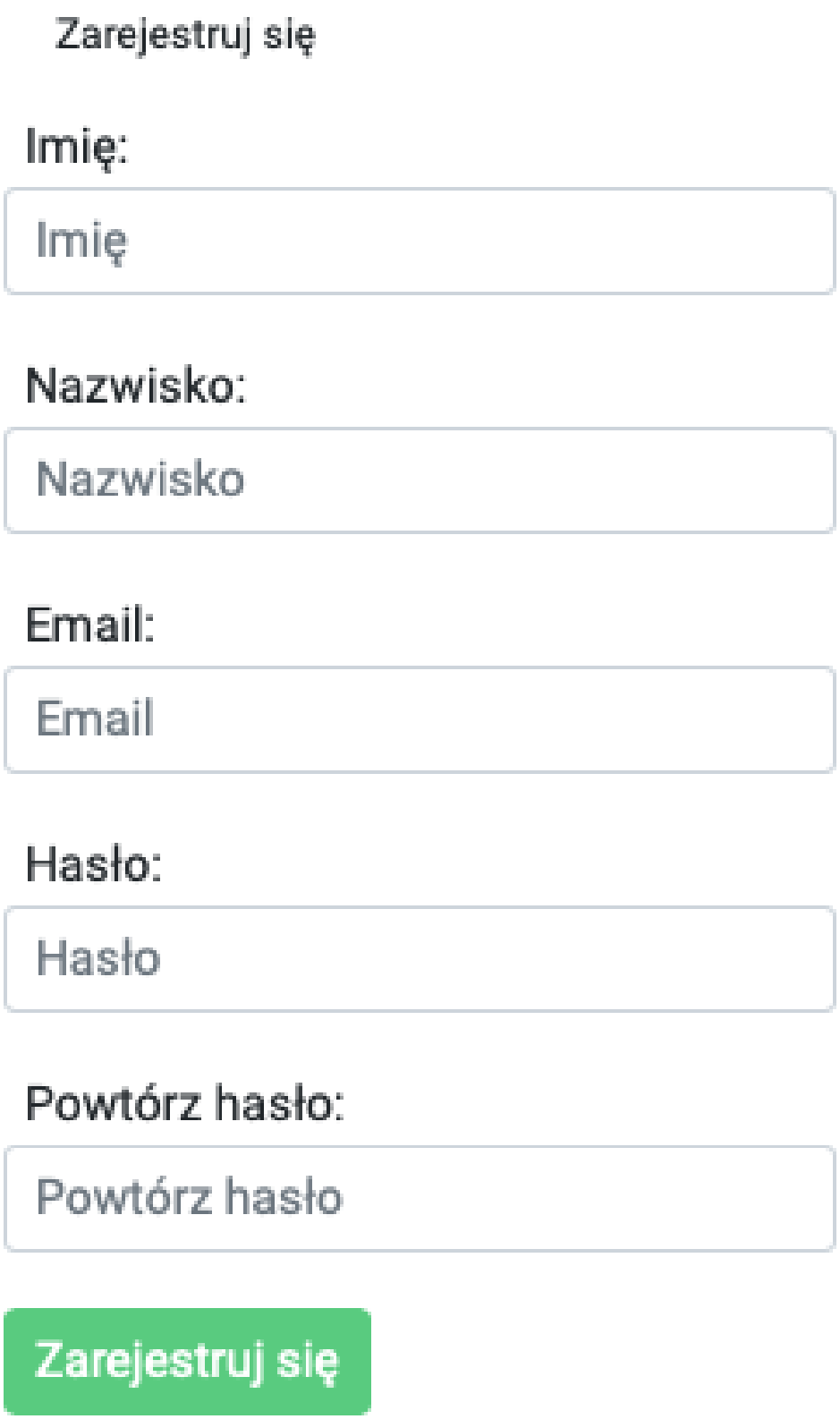
```
export class LoginComponent implements OnInit{
  public user: User;
  isLoggedin = false;
  isLoginFailed = false;
  errorMessage = '';
  roles: string[] = [];

  constructor(private loginService: LoginService,
  private tokenStorage: TokenStorageService, private toastr: ToastrService,
              private router: Router){
    this.user = {id: '', firstName: '', lastName: '',
    role: [] , email: '', password: '', accessToken: ''};
  }

  onSubmit() {
    this.loginService.login(this.user).subscribe(
      data => {
        this.tokenStorage.saveToken(data.accessToken);
        this.tokenStorage.saveUser(data);
        this.tokenStorage.saveUserID(data.id);
        this.isLoginFailed = false;
        this.isLoggedin = true;
        this.roles = this.tokenStorage.getUser().roles;
        this.toastr.success('Logowanie Pomyślne', 'Sukces');
        this.router.navigate(['/']);

      },
      err => {
        this.toastr.error('Logowanie nie powiodło się', 'Błąd');
        this.errorMessage = err.error.message;
        this.isLoginFailed = true;
      }
    );
  }
```

Listing 3: Komponent logowania

3.1.3. Rejestracja

Formularz rejestracji umożliwia użytkownikowi założenie konta. Zasada działania tego formularza po stronie GUI jest analogiczna jak w przypadku logowania.

Rys. 3.3: Formularz rejestracji



3.1.4. Dashboard

Jak widać na ekranie powitalnym, na górze strony znajduje się pasek, który udostępnia przyciski do zalogowani i rejestracji. Jest to dashboard naszej aplikacji. Na początku nie daje on nam dużo funkcjonalności, jednak w momencie, gdy jesteśmy już zalogowani dostajemy dostęp do kolejnych przycisków.

Dwa najważniejsze z nich umożliwiają nam dostęp do podstron, które nie są dostępne przed zalogowaniem. Są to podstrony odpowiedzialne za podsumowanie i zarządzanie spotkaniami użytkownika oraz podstrona umożliwiająca zarządzanie zespołami, w których użytkownik się znajduje

Dostajemy również dostęp do przycisku wylogowania, który pozwala nam w bezpieczny sposób usunąć wszystkie tokeny po skończonej pracy, aby nikt niepożądany nie mógł korzystać z naszego konta bez naszej wiedzy w przypadku, gdy np. odejdziemy od urządzenia.

Przez cały czas widoczny jest skrót nazwy aplikacji - MOS. Ten element także ma swoją funkcję. Po jego naciśnięciu zostaniemy przekierowani z powrotem na stronę główną.

Rys. 3.4: Dashboard po zalogowaniu

3.1.5. Zespół

W aplikacji frontendowej, utworzony został interfejs, reprezentujący zespół, o następujących atrybutach:

```
import { Members } from './members';

export interface Team{
    id: string;
    teamName: string;
    members: Members;
  }
```

Listing 4: Model zespołu

```
export interface Members {
  SCRUM_MASTER: Array<string>;
  TEAM: Array<string>;
  PRODUCT_OWNER: Array<string>;
}
```

Listing 5: Model członków

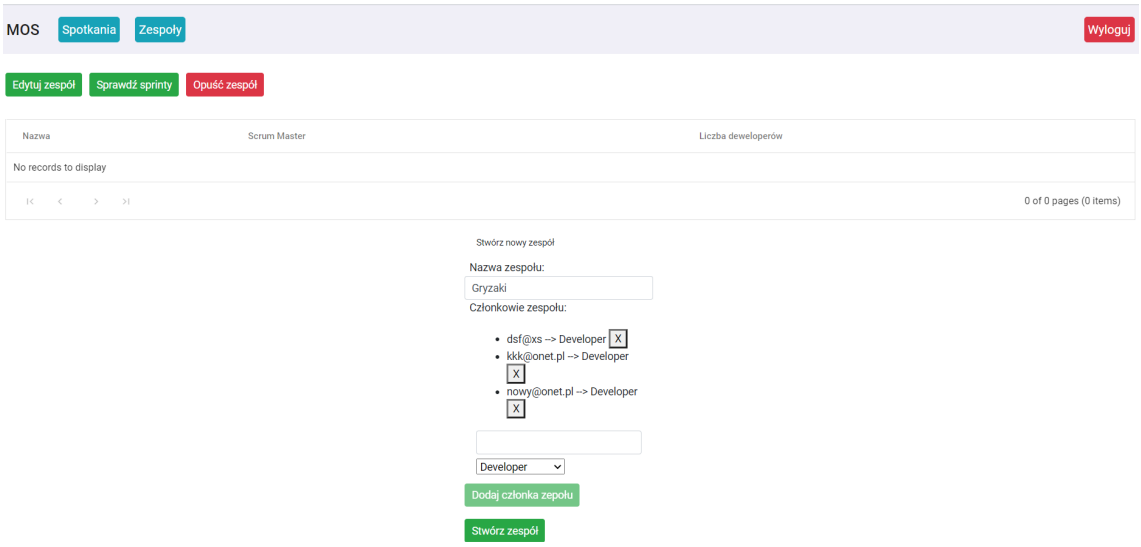
W serwisie CreateTeamService, znajdują się metody wysyłającą zapytania do Backendu, odpowiedzialne za tworzenie, edycje, usuwanie oraz zwracanie zespołów. Parametry zespołu oraz użytkownika są podawane jako argumenty metod.

```
export class CreateTeamService {
  private team: Observable<Team>;
  private teams: Observable<Team[]>;
  private temTeam: any;
  private tem: any;
  constructor(private http: HttpClient) {}

  public createTeam(team: Team): Observable<Team> {
    console.log(team);
    this.http
      .post<Team>('http://localhost:8081/teams', team)
      .toPromise()
      .then((data) => {
        console.log(data);
      });
    return this.team;
  }

  public getAllTeams(): Observable<string> {
    return this.http.get<string>('http://localhost:8081/getAllTeams');
  }

  public getTeam(teamID: string): Observable<Team> {
    return this.http.get<Team>('http://localhost:8081/teams/' + teamID);
  }

  public editTeam(team: Team){
    console.log(team);
    this.http
      .put<Team>('http://localhost:8081/teams/' + team.id, team)
      .toPromise()
      .then((data) => {
        console.log(data);
      });
  }

  public deleteTeam(teamID: string) {
    this.http.delete('http://localhost:8081/teams/' + teamID).toPromise().then();
  }
```

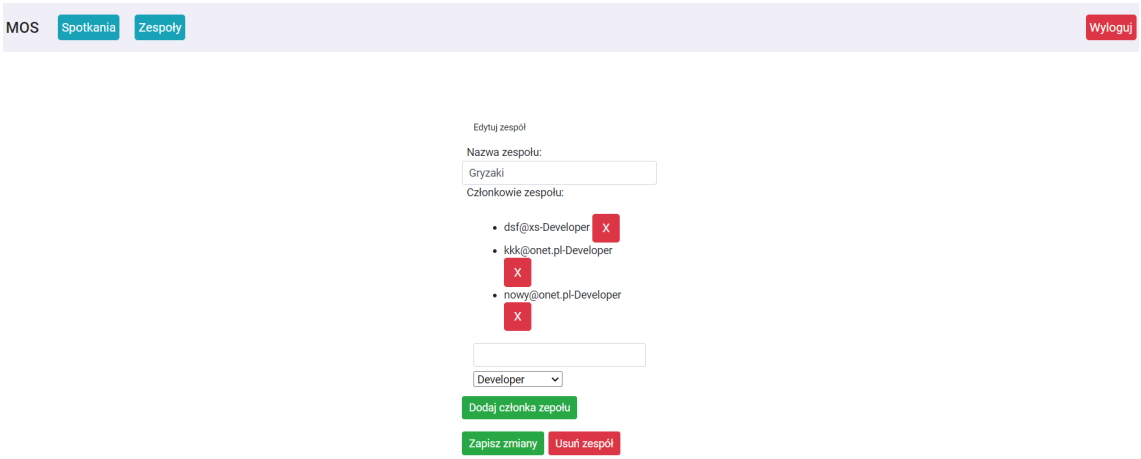
Listing 6: Serwis zespołu

W komponentach TeamComponent, TeamsTableComponent oraz TeamsDetailComponent wywoływane są dane metody z serwisu. Te komponenty odpowiadają za połączenie logiki odpowiedzialnej za większość działań na zespole.

Dla edycji zespołu przewidziany jest osobny komponent, do którego dostęp ma tylko Scrum Master, którego userID sprawdzane jest na podstawie ID użytkownika przechowywanego w sesji.



Rys. 3.5: Serwis zespołu.



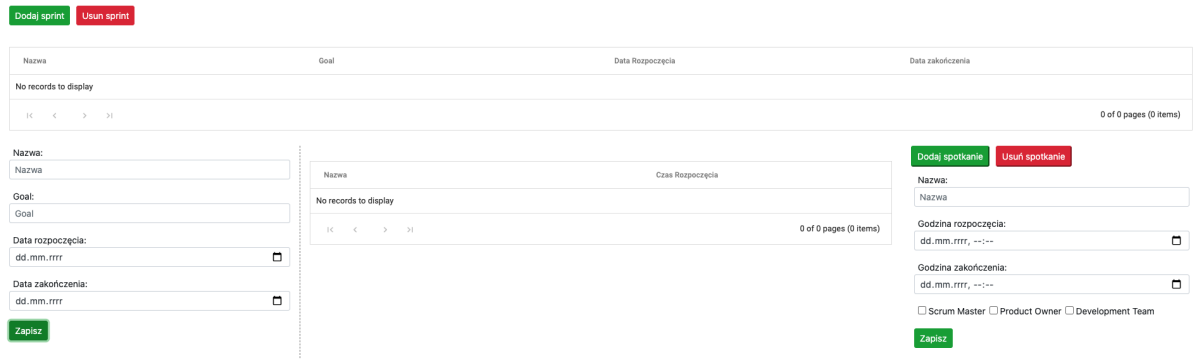
Rys. 3.6: Serwis zespołu.

3.1.6. Sprint

Po wybraniu odpowiedniej opcji w module edycji zespołu, możemy dostać się do podstrony odpowiedzialnej za tworzenie sprintu. Za jej pomocą możemy wykonać wszystkie operacje związane z CRUD'em (eng. Create, Read, Update, Delete), czyli metodami zapisu, odczytu, aktualizacji i usuwania danych. Wszystkie sprinty wyświetlają się w tabeli, która umożliwia filtrowanie i sortowanie, aby w łatwy sposób można było znaleźć interesujący na sprint. Dodawanie i edycja są wykonywane za pomocą formularza znajdującego się pod tabelką. Do usuwania zespołów przeznaczony został konkretny przycisk.

Wszystkie operacje związane z implementacją zmian po stronie serwera zostały zaimplementowane w specjalnym serwisie, który wykorzystuje możliwości Angulara i Javascriptu - Observable i Promise, aby wszystkie zmiany przebiegały płynnie i nie powodowały dyskomfrotu podczas korzystania z aplikacji.

Struktura działania CRUD'u sprintu jest bardzo podobna do CRUD'u zespołu.



Rys. 3.7: CRUD sprintu i spotkania

```
export interface Meeting{
    id: number;
    sprintId: string;
    meetingName: string;
    members: string[];
    startDateTime: Date | string;
    endDateTime: Date | string;
}
```

Listing 7: Interfejs reprezentujący spotkanie

3.1.7. Spotkania

Moduł związany z CRUD'em spotkania został dodany jako część większego modułu związanego ze sprintami. Dlatego wszystkie spotkania można wygodnie dodać w tym samym miejscu w którym edytujemy sprinty. Pozwala to również dodawać i modyfikować spotkania dla wielu sprintów na tej samej podstronie co sprawia, że interfejs jest bardziej przyjemny dla użytkownika.

Podobnie jak w przypadku sprintu, do wyświetlania spotkań została wykorzystana tabelka, a do dodawania i edycji danych wykorzystywany jest dedykowany formularz.

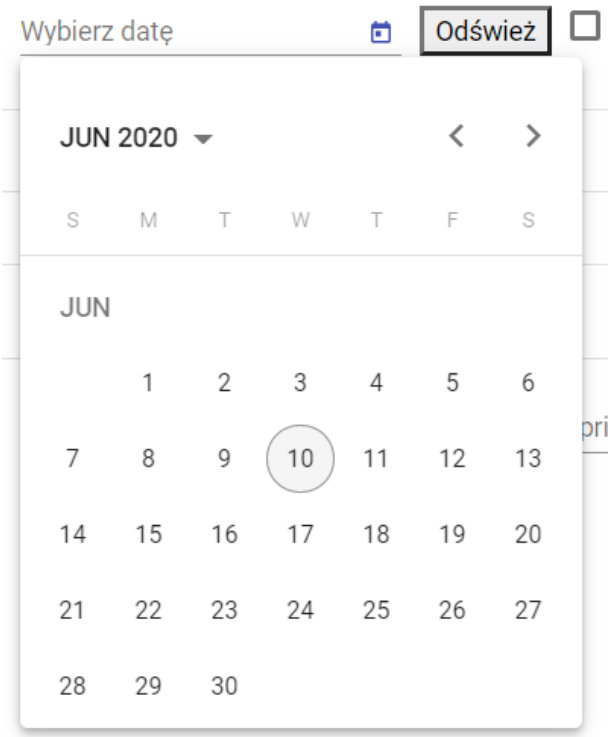
W aplikacji frontendowej, utworzony został interfejs, reprezentujący spotkania, o odpowiednich atrybutach.

W serwisie get-meeting.service, znajdują się metody wysyłającą zapytanie do Backendu, zwracającą wszystkie spotkania danego użytkownika, danego zespołu, oraz danego Sprintu. Parametry zespołu i sprintu (ID) są podane jako argumenty metod, a id użytkownika pobierane są z session Storage. Funkcje zwracają listy obiektów Observable z danymi spotkaniami (8). W komponencie get-meetings.component wywoływane są dane metody z serwisu. Domyślnie wywoływana jest metoda zwracająca spotkania użytkownika, jednak utworzone zostały elementy pozwalające na wybór filtrowania wg zespołu lub sprintu, dzięki którym komponent może wybrać odpowiednią metodę.

Każda z metod w komponencie ma filtrowanie listy spotkań, tak aby zawierała ona tylko spotkania z wybranego dnia. Wybór dnia, został zaimplementowany z wykorzystaniem modułu *Date Picker* z pakietu *Angular Materials*.

```
public loadSprintMeeting(
    teamId: string,
    sprintId: string
  ): Observable<Meeting[]> {
    this.meetingsprint = this.http.get<Meeting[]>(
      API_LINK +
        `users/` +
        window.sessionStorage.getItem('user_id') +
        `/teams/${teamId}/sprints/${sprintId}/meetings`
    );

    return this.meetingsprint;
  }

  public loadUserMeeting(): Observable<Meeting[]> {
    this.metingsuser = this.http.get<Meeting[]>(
      API_LINK +
        'users/' +
        window.sessionStorage.getItem('user_id') +
        '/meetings'
    );

    return this.metingsuser;
  }
  public loadTeamMeeting(teamId: string): Observable<Meeting[]> {
    this.metingsteam = this.http.get<Meeting[]>(
      API_LINK +
        `users/` +
        window.sessionStorage.getItem('user_id') +
        `/teams/${teamId}/meetings`
    );
    return this.metingsteam;
  }
```

Listing 8: get-meeting.service.ts



Rys. 3.8: Wybór daty spotkania

Zaznaczona data przechowywana jest w obiekcie *datachoosen*. Wykorzystuje je funkcja *filter*, która filtruje listę obiektów typu Observable.

```
this.getMeetingsSevice.loadUserMeeting().subscribe((ret) => {
        this.test = ret.filter(mett => {
          const dateOfMeet = new Date(mett.startDateTime);
          const numberofweekmeet = this.getWeekNumber(dateOfMeet);
          const numberofweekchoosen = this.getWeekNumber(this.datachoosen);
          if (numberofweekmeet === numberofweekchoosen){
           return true;
         }else{
          return false;
         }

        });
      });
   }else{
    this.getMeetingsSevice.loadUserMeeting().subscribe((ret) => {
      this.test = ret.filter(mett => {
        const yearmet = new Date(mett.startDateTime).getFullYear();
        const monthmet = new Date(mett.startDateTime).getMonth();
        const daymet = new Date(mett.startDateTime).getDate();


        const year = this.datachoosen.getFullYear();
        // console.log(year)
        const month = this.datachoosen.getMonth();
        // console.log(month)
        const day = this.datachoosen.getDate();
     // console.log('wyb: '+day)

        if (year === yearmet && month === monthmet && day === daymet){
        return true;
       }else{
        return false;
       }
      });

    });
```

Listing 9: Filtrowanie listy zwracanej przez metodę serwisu get-meeting.service

3.1.8. Uruchomienie aplikacji jako PWA

Żeby zmienić aplikację angularową, aplikację PWA, należało zaimplementować w aplikacji Service Worker. Jest to mechanizm który jest uruchamiany w przeglądarce odpowiedzialny między innymi za uruchamianie skryptów w tle, w osobnych wątkach aplikacji oddzielonym od wątku głównego oraz za zarządzaniem pamięcią cache aplikacji.

```
ng add @angular/pwa
```

Listing 10: Dodanie service worker do aplikacji

W pliku ngsw-config.json utworzona została konfiguracja sposobu cache'owania zasobów aplikacji.

```
{
    "$schema": "./node_modules/@angular/service-worker/config/schema.json",
    "index": "/index.html",
    "assetGroups": [{
        "name": "app",
        "installMode": "prefetch",
        "resources": {
            "files": [
                "/favicon.ico",
                "/index.html",
                "/manifest.webmanifest",
                "/*.css", ]
        }
    }, {
        "name": "assets",
        "installMode": "lazy",
        "updateMode": "prefetch",
        "resources": {
            "files": [
                "/assets/**",
                "/*.(eot|svg|cur|jpg|png|webp|gif|otf|ttf|woff|woff2|ani)" ]
        }
    }],
    "dataGroups": [{
        "name": "MoS-api",
        "urls": [
            "http://156.17.41.226:8081/"
        ],
        "cacheConfig": {
            "strategy": "performance",
            "maxAge": "1d",
            "maxSize": 100
        }
```

Listing 11: plik ngsw-config.json

```
    }]
}
```

```
"serviceWorker": true,
"ngswConfigPath": "ngsw-config.json"
```

Listing 13: Włączenie Service Workera w pliku angular.json

Aby użytkownik miał możliwość aktualizacji aplikacji, poprzedzonej odpowiednim komunikatem po odświeżeniu strony, napisano w pliku app.component metodę, używającą klasy SwUpdate która pozwala na aktualizowanie Service Workera.

```
reloadCache(){
    if (this.swUpdate.isEnabled){
      this.swUpdate.available.subscribe(() => {
        if (confirm('Dostępna nowa wersja aplikacji! Czy chcesz dokonać aktualizacji
          window.location.reload();
        }
      });
    }
  }
```

Aby aplikacja PWA, obsługiwała powiadomienia Push-Notification powstał serwis napisany w technologii exspress z wykorzystaniem npm web-push module.

```
WebUI> npm install web-push
```

Listing 14: Instalacja web-push module

Za pomocą tego narzędzia, wygenerowano parę kluczy do uwierzytelnienia komunikacji aplikacji z serwerem do push-notifications. Aplikacja za pomocą klucza publicznego ma dostęp do serwera.

```
let express = require('express');
let bodyParser = require('body-parser');
let cors = require('cors');
let webpush = require('web-push');
let app = express();
app.use(bodyParser.urlencoded({
    extended: false
}));
app.use(bodyParser.json());
app.use(cors());
app.get('/', (req, res) => {
    res.send('This is a push notification server use post');
});
app.post(`/subscribe`, (req, res) => {
    let sub = req.body;
    res.set('Content-Type', 'application/json');
    webpush.setVapidDetails(
        'mailto:example@yourdomain.org',
        "BJas3w-7PyTZl1iqqdsHuOXW1eSFXhpp-Jrh_hz7TwL02iIaDyyL2zAGngTvfKldvGvHuHz2YsE
        "04okdLLP6rfKmSLHHJx44sUZx6NhURh3WqgLOxFqlHg"
    );
    let payload = JSON.stringify({
        "notification": {
            "title": "Przypominacz Spotkań",
            "body": "Masz zbliżające się spotkanie!",

        }
    });
    Promise.resolve(webpush.sendNotification(sub, payload))
        .then(() => res.status(200).json({
            message: 'Notification sent'
        }))
        .catch(err => {
            console.error(err);
            res.sendStatus(500);
        })
})
app.listen(3000, () => {
    console.log('Listening on port 3000...');
});
```

Listing 15: pushserver.js - Backend wysyłający push-notification

W Aplikacji angularowej utworzono serwis za pomocą którego frontend,może wysyłać powiadomienia.

```
import { catchError } from 'rxjs/operators';
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { PUSH_LINK } from 'src/assets/connection';

@Injectable({
  providedIn: 'root',
})
export class NotificationsService {
  public notficationURL = PUSH_LINK + `subscribe`;
  constructor(private _http: HttpClient) {}

  postSubscription(sub: PushSubscription) {
    console.log('sent notification');
    return this._http
      .post(this.notficationURL, sub)
      .pipe(catchError(this.handlError));
  }
  handlError(handlError: any): any {
    throw new Error('Method not implemented.');
  }
}
```

Listing 16: notification.service.ts

3.2. Backend

3.2.1. Rejestracja

Żeby korzystać z aplikacji musimy mieć konto, dla tego służy rejestracja. Rejestracja jest realizowana w następny sposób:backend otrzymuje dane z frontendu, sprawdza czy tego użytkownika nie ma w naszej bazie danych. Sprawdzenie odbywa się za pomocą podanej poczty elektronicznej, jeżeli taka już istnieje, to jest wysłany błąd do frontendu, w przeciwnym przypadku dodajemy nowego użytkownika do naszej bazy danych.

```
@PostMapping("/signup")
public ResponseEntity<?> registerUser(@Valid @RequestBody SignupRequest signUpReques
     if (userRepository.existsByEmail(signUpRequest.getEmail())) {
        return ResponseEntity
                .badRequest()
                .body(new MessageResponse("Error: User with email: " + signUpRequest
    }
    if (userRepository.existsByEmail(signUpRequest.getEmail())) {
        return ResponseEntity.badRequest()
                .body(new MessageResponse("Error: Email is already in use!"));
    }
    User user = new User(signUpRequest.getFirstName(),
            signUpRequest.getLastName(),
            signUpRequest.getEmail(),
            encoder.encode(signUpRequest.getPassword()));
    Set<String> strRoles = signUpRequest.getRoles();
    Set<Roles> roles = new HashSet<>();
    if (strRoles == null) {
        Roles userRole = roleRepository.findByName(ERole.ROLE_USER)
         .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
        roles.add(userRole);
    } else {
        strRoles.forEach(role -> {
            switch (role) {
                case "admin":
                Roles adminRole = roleRepository.findByName(ERole.ROLE_ADMIN)
                .orElseThrow(() -> new RuntimeException("Error: Role is not found.")
                roles.add(adminRole);
                    break;
                case "mod":
                Roles modRole = roleRepository.findByName(ERole.ROLE_MODERATOR)
                .orElseThrow(() -> new RuntimeException("Error: Role is not found.")
                roles.add(modRole);
                    break;
                default:
                Roles userRole = roleRepository.findByName(ERole.ROLE_USER)
                .orElseThrow(() -> new RuntimeException("Error: Role is not found.")
                roles.add(userRole);
            }
        });
    }
    user.setRoles(roles);
    userRepository.save(user);
    LoginRequest loginRequest = new LoginRequest();
    loginRequest.setEmail(signUpRequest.getEmail());
    loginRequest.setPassword(signUpRequest.getPassword());
    return login(loginRequest);
}
```

Listing 17 AuthController.java (Rejestracja)

3.2.2. Autoryzacja

W aplikacji jest możliwość logowania oraz wylogowania. Ze strony backendu to jest realizowane za pomocą JWT(JSON Web Token). Ten sposób realizacji jest dość często wykorzystany w praktyce. Jedną z wad JWT jest to, że go nie można po prostu usunąć ze strony backendu. Więc sposób w jaki jest on unieważniony jest następujący. W naszej aplikacji skorzystaliśmy z Redis. Redis - to baza danych typu klucz-wartość, która przechowuje dane w pamięci RAM. W naszej aplikacji to służy jako blacklista JWT tokenów, tak zwany cache. Działa to w następny sposób:
1. Użytkownik wprowadza swoje dane, w przypadku logowania sprawdza się czy jest ten użytkownik w bazie danych.
2. Po poprawnym logowaniu, zostaje utworzony token na podstawie danych użytkownika, który ma określony czas życia(w naszym przypadku ustaliliśmy 24 godziny, ale można ustalać dowolny).
3. Dalej sprawdzamy czy tego tokenu nie zawiera nasza blacklista, jeżeli nie zawiera, to autoryzacja kończy się sukcesem, w przeciwnym przypadku użytkownik będzie przekierowany na stronę logowania.
4. Po wylogowaniu ten token zostaje dodany do naszej blacklisty oraz nie może już być wykorzystany do autoryzacji.

```
@PostMapping("/signin")
public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest loginRequ
    return login(loginRequest);
}
private ResponseEntity<?> login(LoginRequest loginRequest){
    Authentication authentication = authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(loginRequest.getEmail(), loginRe

    SecurityContextHolder.getContext().setAuthentication(authentication);
    String jwt = jwtUtils.generateJwtToken(authentication);

    UserDetailsImpl userDetails = (UserDetailsImpl) authentication.getPrincipal();
    List<String> roles = userDetails.getAuthorities().stream()
            .map(item -> item.getAuthority())
            .collect(Collectors.toList());

    return ResponseEntity.ok(new JwtResponse(jwt,
            userDetails.getId(),
            userDetails.getEmail(),
            roles));
}
```

Listing 18: AuthController.java(logowanie)

3.2.3. Zespół

W aplikacji backendowej utworzony został POJO Team.java, reprezentujący spotkania, o odpowiednich atrybutach. Nie wszystkie te atrybuty wykorzystane do przesyłania na frontend. Ale wszystkie są zachowane w bazie danych. Dodatkowo dla ułatwiania przekazania danych do frontendu stworzony DTO i mapper dla przetworzenia.

```
package com.scruum.mastersofscruum.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import lombok.Data;
import org.springframework.data.annotation.Id;
import java.util.List;
import java.util.Map;

@Data
public class Team {
    @Id
    private String id;
    private String teamName;
    private Map<Role, List<String>> members;
    @JsonIgnore
    private List<String> sprintList;
    @JsonIgnore
    private List<Meeting> meetingList;
    /**
        Getters and Setters
    **/
}
```

Listing 19: Team.java

Kontroler TeamController.java zawiera metody które nasłuchują na zapytania http i zwracają odpowiednie reqesty do frontendu oraz współpracuje z bazą danych, jeśli na to jest potrzeba.

Komunikacja z bazą danych zachodzi poprzez serwis TeamService.java, TeamServiseImpl.java oraz repositori TeamRepository.java, gdzie są formułowane zapytania.

```java
package com.scruum.mastersofscruum.service.Impl;

import com.scruum.mastersofscruum.dto.TeamDto;
import com.scruum.mastersofscruum.model.Team;
import com.scruum.mastersofscruum.repository.TeamRepository;
import com.scruum.mastersofscruum.service.TeamService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;
import org.springframework.web.server.ResponseStatusException;
import java.util.Optional;

@Service
public class TeamServiceImpl implements TeamService {

    private final TeamRepository teamRepository;
    private final MongoTemplate mongoTemplate;

    @Autowired
    public TeamServiceImpl(TeamRepository teamRepository,
                           MongoTemplate mongoTemplate) {
        this.teamRepository = teamRepository;
        this.mongoTemplate = mongoTemplate;
    }
    public TeamRepository teamRepository() {
        return teamRepository;
    }

    @Override
    public Team updateTeam(Team team, TeamDto teamDto) {

        team.setTeamName(teamDto.getTeamName());
        team.setMembers(teamDto.getMembers());
        team = teamRepository.save(team);
        return team;
    }

    @Override
    public Team findById(String teamId) throws ResponseStatusException {
        Optional<Team> optionalTeam = teamRepository.findById(teamId);
        if (optionalTeam.isEmpty()) {
            throw new ResponseStatusException(HttpStatus.NOT_FOUND);
        }
        return optionalTeam.get();
    }
}
```

Listing 20: TeamServiceImpl.java

3.2.4. Sprint

W aplikacji backendowej utworzony został POJO Sprint.java, reprezentujący sprint, o odpowiednich atrybutach. Wszystkie zachowują się w bazie i są połączone z zespołem tak, że Team POJO ma w sobie listę ID połączonych z nią sprintów. Dodatkowo dla ułatwiania przekazania danych do frontendu stworzony DTO i mapper dla przetworzenia.

```
package com.scruum.mastersofscruum.model;

import com.fasterxml.jackson.annotation.JsonFormat;
import io.swagger.v3.oas.annotations.media.Schema;
import lombok.Data;
import org.springframework.data.annotation.Id;
import java.time.LocalDate;

@Data
public class Sprint {
    @Id
    private String id;
    private String sprintName;
    private String goal;
    @JsonFormat(pattern = "yyyy-MM-dd")
    @Schema(pattern = "yyyy-MM-dd")
    private LocalDate startDate;
    @JsonFormat(pattern = "yyyy-MM-dd")
    @Schema(pattern = "yyyy-MM-dd")
    private LocalDate endDate;
    /**
    Getters and Setters
    **/
}
```

Listing 21: Sprint.java.java

Kontroler SprintController.java zawiera metody które nasłuchują na zapytania http i zwraca odpowiednie reqesty do frontendu oraz współpracuje z bazą danych, jeśli na to jest potrzeba.

W serwisie SprintService.java oraz SprintRepository.java formułowane zapytania do bazy danych.Podczas dodawania lub usuwania sprintu, po stronie bazy danych zmienia się zapis w tabeli team, ponieważ zespół zawiera listę id sprintów do niej należących.

```java
@Override
public boolean deleteById(String teamId, String sprintId) {
    Optional<Team> optionalTeam = teamRepository.findById(teamId);
    if (optionalTeam.isPresent()) {
        Team team = optionalTeam.get();
        if (sprintRepository.existsById(sprintId) &&
                    team.getSprintList().contains(sprintId)) {
            team.getSprintList().remove(sprintId);
            team = teamRepository.save(team);
            sprintRepository.deleteById(sprintId);
            if (!teamRepository.existsById(sprintId))
                return true;
         }
    }
    return false;
}
@Override
public List<Sprint> findByTeamId(String teamId) {
    Optional<Team> optionalTeam = teamRepository.findById(teamId);
    if (optionalTeam.isPresent()) {
        Team team = optionalTeam.get();
        List<Sprint> list = new ArrayList<>();
        if(team.getSprintList() != null) {
            for (String id : team.getSprintList()) {
                Optional<Sprint> optionalSprint = sprintRepository.findById(id);
                if (optionalSprint.isPresent()) {
                    Sprint sprint = optionalSprint.get();
                    list.add(sprint);
                }
            }
            return list;
        }
    }
    return new ArrayList<>();
}
@Override
public Sprint save(String teamId, Sprint sprint) {
    Optional<Team> optionalTeam = teamRepository.findById(teamId);
        if (teamRepository.existsById(teamId)) {
            sprint.setId(null);
            sprint = sprintRepository.save(sprint);
            Team team = optionalTeam.get();
            Team.getSprintList().add(sprint.getId());
            teamRepository.save(team);
            return sprint;
    }
    return null;
}
```

Listing 22: SprintServiceImpl.java (najważniejsze funkcje)

3.2.5. Spotkania

W aplikacji backendowej utworzony został POJO Meeting.java, reprezentujący spotkania, o odpowiednich atrybutach. Wszystkie zachowują się w bazie danych jako element listy spotkań należących do zespołu.

```
package com.scruum.mastersofscruum.model;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import io.swagger.v3.oas.annotations.media.Schema;
import lombok.*;
import javax.validation.constraints.NotNull;
import java.time.LocalDateTime;
import java.util.List;

@Getter
@Setter
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Data
public class Meeting {
    private int id;
    @NotNull
    private String meetingName;
    @NotNull
    @JsonFormat(pattern = "yyyy-MM-dd'T'HH:mm")
    @Schema(pattern = "yyyy-MM-dd'T'HH:mm")
    private LocalDateTime startDateTime;
    @JsonFormat(pattern = "yyyy-MM-dd'T'HH:mm")
    @Schema(pattern = "yyyy-MM-dd'T'HH:mm")
    private LocalDateTime endDateTime;
    private String sprintId;
    private List<Role> members;

    @Override
    public String toString() {
        Gson gson = new GsonBuilder().setPrettyPrinting().create();
        return gson.toJson(this);
    }
}
```

Listing 23: Meeting.java

Kontroler MeetingController.java zawiera metody które nasłuchują na zapytania http i zwraca odpowiednie reqesty do frontendu oraz współpracuje z bazą danych, jeśli na to jest potrzeba.

W serwisie MeetingService.java i MeetingServiceImpl.java formułowane zapytania do bazy danych. Jak napisano wcześniej, spotkania przechowują się w zespole, dlatego wszystkie zmiany i odwołania do Meeting powodują pracę z Team.

```
package com.scruum.mastersofscruum.service;

import com.scruum.mastersofscruum.model.Meeting;
import com.scruum.mastersofscruum.model.User;
import java.util.List;

public interface MeetingService {

    Meeting save(String teamId, Meeting meeting);

    Meeting update(String teamId, Meeting meeting);

    List<Meeting> findAll(String teamId);

    boolean deleteByMeetingName(String teamId, Integer meetingId);

    Meeting findById(String teamId, Integer meetingId);

    List<User> findMembersByMeetingId(String teamId, Integer meetingId);

    List<Meeting> getMeetings(String userId);

    List<Meeting> findMeetingsByUserId(String userId);

    List<Meeting> getMeetingsByUserIdAndTeamIdAndSprintId(String userId,
                                                          String teamId, String sprintId);

    List<Meeting> getMeetingsByUserIdAndTeamId(String userId, String teamId);
}
```

Listing 24: MeetingService.java

Z powodu możliwości tworzenia dużej ilości spotkań, zostały zrealizowane filtry. Dla ich wykorzystania istnieją kontrolery w pliku FilterController.java. W celu zmniejszenia obciążenia na fronend, backend i sieć, większość danych filtruje się i obcina się po stronie bazy danych. Zapytania do bazy danych formułowane w MeetingSevice.java i MeetingServiceIml.java

```java
package com.scruum.mastersofscruum.controller;
/**
Imports
**/
@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
public class FilterController {
    @Autowired
    TeamService teamService;
    @Autowired
    MeetingService meetingService;

    @GetMapping(value = "users/{userId}/teams",
            produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<Team>> getUserTeams(@PathVariable String userId){
        return new ResponseEntity<List<Team>>(teamService.teamRepository()
                .findTeamByMember(userId), HttpStatus.OK);
    }
    @GetMapping(value = "users/{userId}/meetings",
            produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<Meeting>> getMeetingByUserId(
                                                @PathVariable String userId){
        return new ResponseEntity<List<Meeting>>(meetingService
                .findMeetingsByUserId(userId), HttpStatus.OK);
    }
    @GetMapping(value="users/{userId}/teams/{teamId}/sprints/{sprintId}/meetings")
    public ResponseEntity<List<Meeting>> getMeetingByUserIdANDTeamIdAndSprintId(
            @PathVariable String userId, @PathVariable String teamId,
            @PathVariable String sprintId){
        return new ResponseEntity<List<Meeting>>(
                meetingService.getMeetingsByUserIdAndTeamIdAndSprintId(userId,
                teamId,sprintId), HttpStatus.OK);
    }
    @GetMapping(value = "users/{userId}/teams/{teamId}/meetings")
    public ResponseEntity<List<Meeting>> getMeetingByUserIdAndTeamId(
            @PathVariable String userId, @PathVariable String teamId){
        return new ResponseEntity<List<Meeting>>(
                meetingService.getMeetingsByUserIdAndTeamId(userId,teamId),
                HttpStatus.OK);
    }
    @Deprecated
    @GetMapping(value = "users/{userId}/meetings$filter",
            produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<?> filterMeetings(@PathVariable String userId,
                                    @RequestParam Map<String,String> allParams){
        return new ResponseEntity<>(allParams.entrySet(), HttpStatus.OK);
    }
}
```

Rozdział 4

Przebieg prac nad projektem

4.1. Sprint 0

Jest to pierwsze spotkanie z Panem profesorem. Podczas tego spotkania dobraliśmy się w grupy projektowe i wybraliśmy temat. Spotkanie odbyło się 27.02.2020.

Raport Projektu Zespołowego

Data spotkania: 27.02.2020
Miejsce spotkania: Politechnika Wrocławska

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, Mgr. Inż. Mateusz Żurawski
   - Członkowie grupy : Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
   - Dr. Rogaliński przedstawił cele oraz zasady zaliczenia projektu zespołowego. Opowiedział o przebiegu pracy nad projektem oraz o funkcji kierownika grupy projektowej.
   - Prezentacja na temat Konferencji Projektów Zespołowych
   - Pan Mateusz Żurawski z firmy „Comarch" przedstawił temat oraz założenia projektu. Wyjaśnił skąd potrzeba danej aplikacji, założenia oraz wizje dotyczące efektu finalnego.

3. **Podsumowanie**
   - Aplikacja odciążyłaby Scrum Mastera z wielu niepotrzebnych dla człowieka czynności, takich jak przypominanie członkom zespołu o terminach spotkania
   - Aplikacja wymaga tego by była dopasowana do wszystkich użytkowników, w tym celu powinna być obsługiwana na systemach zarówno Ios jak i Android, dlatego należy ją napisać jako tzw. Aplikacje hybrydową.

4. **Zadania do Wykonania**
   - Michał Juszczyk – Utworzenie projektu na platformach Slack oraz Taiga, oraz zaproszenie do nich wszystkich uczestników projektu
   - Maciej Jaroński – Napisanie raportu ze spotkania
   - Wszyscy członkowie – Zapoznanie się z materiałami udostępnionymi przez Pana Mateusza Żurawskiego

Rys. 4.1: Raport 27.02

4.2. Sprint 1

Sprint trwał od 06.03.2020 do 19.03.2020
Zadania do Realizacji:

- Prezentacja na zajęcia PWR (z powodu koronawirusa zmienione we wstęp tego raportu)
- Wybór rozwiązania Backendowego
- Wybór technologii hybrydowej
- Wymyślić nazwę zespołu

Raport Projektu Zespołowego

Data spotkania: 06.03.2020
Miejsce spotkania: Comarch

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal
   - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
   - Pan Jan Niekowal opowiedział o Scrumie oraz przedstawił przykład jak to jest wykorzystane w firmie Comarch.
   - Powiedziano o User history oraz Scrum Poker.
   - Pan Jan Niekowal opowiedział o podstawowych wymagania aplikacji.
   - Zostały omówione zadania do wykonania.

3. **Podsumowanie**
   - Utworzone nowe historyjki w środowisku Taiga.io:
     1) Prezentacja na zajęcia na PWr.
     2) Wybór rozwiązania BE.
     3) Wybór rozwiązania aplikacji hybrydowej.
     4) Nazwa zespołu.
   - Otrzymano wiedzę o Scrumie oraz User history.
   - Dowiedziano o podstawowych wymaganiach aplikacji.

4. **Zadania do Wykonania**
   - Vladyslav Lutsenko – Napisanie raportu ze spotkania
   - Wszyscy członkowie – Wymyślenie nazwy dla zespołu. Zapoznanie się z różnymi narzędziami dla tworzenia aplikacji hybrydowej oraz BE, podejmowanie decyzji z jakich z tych narzędzi będzie korzystał zespół. Przygotowanie prezentacji na zajęcia.

Rys. 4.2: Raport 06.03

Raport Projektu Zespołowego

Data spotkania: 16.03.2020
Miejsce spotkania: Discord – konferencja online

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, Mgr. Inż. Mateusz Żurawski
   - Członkowie grupy : Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
   - Razem z Doktorem Rogalińskim próbowaliśmy ustalić założenia aplikacji
   - Pan Niekowal przedstawił prawdopodobną formę aplikacji (PWA)
   - Ustaliliśmy wstępne informacje dotyczące raportu
   - Wspólnie ustaliliśmy termin następnego spotkania

3. **Podsumowanie**
   - Ze względu na sytuację, zamiast prezentacji wykonamy raport opisujący Scrum oraz zawierający założenia aplikacji
   - Do czwartku powinniśmy zdecydować się na konkretne technologie
   - Następne spotkanie odbędzie się w czwartek (19.03) o 16:00

4. **Zadania do Wykonania**
   - Michał Juszczyk – Stworzenie notatki ze spotkania
   - Wszyscy członkowie – Wybranie technologii i rozwinięcie tematu PWA

Rys. 4.3: Raport 16.03

4.3. Sprint 2

Sprint trwał od 19.03.2020 do 02.04.2020
Zadania do Realizacji:

- Znaleźć możliwość wystawienia aplikacji na internet
- Instalacja dockera na serwerze
- Uruchomienie obrazu dockerowego bazy MongoDB
- Uruchomienie aplikacji Spring boot łączącą się z bazą danych MD i wystawiającą dane po REST Api
- Stworzenie projektu Angulara
- Podsumować wiedzę o Scrumie

Raport Projektu Zespołowego

Dara spotkania: 19.03.2020
Miejsce spotkania: serwis rozmów głosowych «Discord»

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal, Mgr. Inż. Mateusz Żurawski
   - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg spotkania:**
   - Pan Jan Niekowal zaakceptował zrobienie zadań omówionych na poprzednim sprincie.
   - Zostały omówione i przyjęte technologie na których powstanie projekt.
   - Zostały omówione zadania do wykonania.

3. **Podsumowanie:**
   - Zatwierdzone zrobienie poprzednich historyjek w środowusku Taiga.io.
   - Utworzone nowe historyjki w środowisku taiga:
     1. Znaleźć możliwość wystawienia aplikacji na internet
     2. Instalacja docker'a na serwerze
     3. Urochomienie obrazu docker'owego bazy MongoDB
     4. Urochomienie aplikacji Spring boot, łączącą się z bazą danych MongoDB i wystawiającą dane po REST API
     5. Stworzenie projektu Angulara
     6. Przygotowanie wiedzy o Scrum'ie

4. **Zadania do wykonania:**
   - Nikita Stepanenko: napisanie raportu, urochomienie aplikacji Spring boot, ustalenie serwera od politechniki
   - Vladyslav Lutsenko: urochomienie aplikacji Spring boot, ustalenie serwera od politechniki
   - Maciek Jaroński: instalacja docker'a na serwerze
   - Łukasz Handschuh: urochomienie obrazu docker'owego bazy MongoDB
   - Michał Juszczyk: stworzenie projektu Angulara
   - Wszyscy człąkowie: przygotowanie wiedzy o Scrum'ie

Rys. 4.4: Raport 19.03

Notatka ze spotkania online Projektu Zespołowego „Masters of Scruum”

Data: 23.03.2020

Uczestniczący:

- Opiekunowie Projekut: Dr. Inż. Paweł Rogaliński, Mgr. Inż. Mateusz Żurawski
- Członkowie Grupy : Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko

Przebieg spotkania:

Odbyło się krótkie spotkanie typu „Daily”. Członkowie grupy kolejno zdali relacje ze zrealizowanych zadań które zostały im przydzielone oraz jakie mają plany dalsze względem ich realizacji. Opiekun Mgr. Mateusz Żurawski udzielił poszczególnych rad do danych zadań

Rys. 4.5: Raport 23.03

Raport Projektu Zespołowego

Data spotkania: 26.03.2020
Miejsce spotkania: serwis rozmów głosowych «Discord»

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal, Mgr. Inż. Mateusz Żurawski
   - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
   - Każdy opowiedział o tym co udało się zrobić oraz co nie.
   - Pan Mateusz Żurawski zapytał o kilka rzeczy, których się udało zrobić.

3. **Podsumowanie**
   - Połowa zrobionych zadań:
     1) Zainstalowany serwer.
     2) Zainstalowany docker na serwerze.

4. **Zadania do Wykonania**
   - Vladyslav Lutsenko – Napisanie raportu ze spotkania. Uruchomienie aplikacji Spring boot.
   - Nikita Stepanenko - Uruchomienie aplikacji Spring boot.
   - Łukasz Handschuh: urochomienie obrazu docker'owego bazy MongoDB.
   - Michał Juszczyk: stworzenie projektu Angulara
   - Wszyscy muszą przygotować wiedzę o Scrumi'e.

Rys. 4.6: Raport 26.03

4.4. Sprint 3

Sprint trwał od 02.04.2020 do 16.04.2020
Zadania do Realizacji:

- Mechanizm logowania
- Konfiguracja zespołu
- Uruchomienie aplikacji PWA
- Rejestracja użytkownika

Raport Projektu Zespołowego

Data spotkania: 02.04.2020
Forma spotkania: Konferencja poprzez Discord

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal, pracownik firmy Comarch Mateusz Żurawski,
   - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
   - Spotkanie miało formę demo oraz Sprint Review.
   - Wszyscy członkowie przedstawili pracę wykonaną podczas sprintu, która została oceniona przez opiekunów.
   - Zostały dodane nowe historyjki do backlogu.
   - Został zaplanowany następny sprint.

3. **Podsumowanie**
   - Zaktualizowanie historyjek w środowisku Taiga.io.
   - Zaplanowanie i rozpoczęcie nowego sprintu

4. **Zadania do Wykonania**
   - Michał Juszczyk, Łukasz Handschuh – Etapy dotyczące aplikacji angularowej z historyjek ze sprintu.
   - Nikita Stepanenko i Vladyslav Lutsenko – Etapy dotyczące aplikacji Java z historyjek ze sprintu.
   - Maciej Jaroński – Zdobycie informacji o technologii PWA i utworzenie prostej aplikacji.
   - Michał Juszczyk – Przygotowanie notatki.

Rys. 4.7: Raport 02.04

Raport Projektu Zespołowego

Data spotkania: 06.04.2020
Forma spotkania: Konferencja poprzez Discord

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal, pracownik firmy Comarch Mateusz Żurawski,
   - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
   - Spotkanie miało formę daily.
   - Wszyscy członkowie przedstawili pracę wykonaną do tego momentu od początku sprinta.
   - Zostało powiedziano o planach do następnego daily.

3. **Podsumowanie**
   - To spotkanie miało na celu podzielić się informacjami i planami z kolegami;
   - To spotkanie miało na celu uzyskanie informacji od kolegów, które mogą się przydać w ciągu pracy.

Rys. 4.8: Raport 06.04

Notatka ze spotkania online Projektu Zespołowego „Masters of Scruum”
Data: 9.04.2020
Uczestniczący:
• Opiekunowie Projektu: Dr. Inż. Paweł Rogaliński, Mgr. Inż. Jan Niekowal
• Członkowie Grupy : Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko

Przebieg spotkania:
Odbyło się krótkie spotkanie typu „Daily”. Członkowie grupy kolejno zdali relacje ze zrealizowanych zadań które zostały im przydzielone oraz jakie mają plany dalsze względem ich realizacji. Opiekun Mgr. Jan Niekowal udzielił poszczególnych rad do danych zadań. Większość uwag poświęcone były standardowi służącemu autoryzacji OAuth 2.0.

Rys. 4.9: Raport 9.04

4.5. Sprint 4

Sprint trwał od 16.04.2020 do 30.04.2020
Zadania do Realizacji:

- Autoryzacja przy użyciu OAuth
- Operacje CRUD na zespole
- CRUD sprintu

Raport Projektu Zespołowego

Data spotkania: 16.04.2020
Miejsce spotkania: serwis rozmów głosowych «Discord»

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal, Mgr. Inż. Mateusz Żurawski
   - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
   - Został skończony Sprint 3.
     Każdy przeprowadził demo tego co zrobił:
     Task #15 Mechanizm logowania: Michał oraz Vladyslav
     Task #12 Uruchomienie aplikacji PWA: Maciek
     Task #13 Rejestracja użytkownika: Michał oraz Vladyslav
     Task #18 Konfigurowanie zespołu: Nikita oraz Łukasz
   - Został utworzony Sprint 4. Pan Mateusz razem z Panem Janem utworzyli nowe taski do zrobienia:
     #35 Autoryzacja przy użyciu OAuth2.0
     #36 Operacje CRUD na zespole
     #17 CRUD sprintu
     #33 Zaplanowanie spotkania w Sprincie.

   - Została przeprowadzona Retrospektywa. Wynik:
     (+)
     Udzielanie pomocy w trakcie pojawienia się problemów, wzajemna pomoc (8)
     dobrze podzielone zadania (każdy wiedzial co robic) (1)
     lepsze drugie demo (1)
     działa połączenie e2e
     Wykonywane zadanie dostarczyło nowych umiejętności
     (-)
     Robione w ostatnie 4 dni, zbyt późne połączenie backadnu i frontendu (6)
     - przeniesienie daily na wtorek
     - wczesniejsze rozpoczęcie pracy, coś zrobione w 1 tydzień
     słaba komunikacja (1)
     - omówienie koncepcji
     - uzgadnianie podejścia do realizowanego zadania
     Początkowo była trudność ze znalezieniem odpowiednich materiałów (1)
     Ograniczony czas w związku ze świętami
     Nie wszystkie user stories zrobione „do końca", pare demo remarków
     - nowy user story
     - dopracować PWA ;)
     Problemy z leagacy Oauth2, brak OAuth
     - nowy user story

Rys. 4.10: Raport 16.04

Raport Projektu Zespołowego

Data spotkania: 23.04.2020
Forma spotkania: Konferencja poprzez Discord Grupa:
Masters of Scruum

1. **Obecni na spotkaniu:**
    - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal
    - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:** • Spotkanie miało formę Refinementu.
    - Każdy z członków opowiedział o postępach historyjek, nad którymi pracuje.
    - Zdecydowano o braku zmian w backlog'u.
    - Wyjaśniono sprawę z wysyłaniem notatek.
    - Zaproponowano technologię, którą ewentualnie można użyć, aby rozwinąć aplikację.

3. **Podsumowanie**
    - Product owner zdecydował o zostawieniu backlog'u w obecnym stanie.
    - Upewniono się, że praca w sprincie przebiega bez opóźnień.

4. **Zadania do Wykonania**
    - Michał Juszczyt – Napisanie raportu ze spotkania.
    - Nikita Stepanenko i Vladyslav Lutsenko – kontynuacja pracy nad utworzeniem komponentów oraz tokenów po stronie aplikacji.
    - Michał Juszczyk, Łukasz Handschuh, Maciej Jaroński – Kontynuacja pracy nad tworzeniem komponentów potrzebnych do obsługi operacji na zespole oraz na Scrumowych spotkaniach.

Rys. 4.11: Raport 23.04

Raport Projektu Zespołowego

Data spotkania: 28.04.2020
Forma spotkania: Konferencja poprzez Discord

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal
   - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
   - Spotkanie miało formę weekly(podsumowanie wykonanego).
   - Wszyscy członkowie przedstawili pracę wykonaną do tego momentu od początku sprinta.
   - Zostało powiedziano o planach do Demo.

3. **Podsumowanie**
   - Zostali omówione bieżące problemy dotyczące CRUD i autoryzacji
   - Utworzone CRUD backend dla zespołu i sprintu - task#17, #36
   - Utworzone CRUD frontend dla zespołu i sprintu - task#17, #36
   - Zrealizowane autoryzacja po stronie backend - task #35
4. **Zadania do Wykonania**
   - Nikita Stepanenko -Napisanie raportu ze spotkania
   - Nikita Stepanenko i Vladyslav Lutsenko - CRUD spotkania po stronie backendu
   - Michał Juszczyk - dołączenie CRUD sprintu ze strony frontendu do backendu - task #17
   - Łukasz Hadschuh - dołączenie CRUD zespołu ze strony frontendu do backendu - task # 36
   - Vladyslav Lutsenko - dopracowanie autoryzacji po stronie backandu - task #35
   - Maciej Jaroński - dołączenie JWT po stronie frontendu - task #35.

Rys. 4.12: Raport 28.04

4.6. Sprint 5

Sprint trwał od 30.04.2020 do 14.05.2020
Zadania do Realizacji:

- Dodawanie użytkownika do zespołu ze zbioru użytkowników aplikacji
- Zaplanowanie spotkania w sprincie

30.04.2020

NOTATKA Z PROJEKTU ZESPOŁOWEGO

1. PLAN SPOTKANIA

Dnia 30.04.2020 odbyło się spotkanie typu Demo. Przedstawiono na nim efekty pracy bieżącego sprintu.

A. Obecni na spotkaniu

i. Opiekunowie projektu: Mgr. Inż Mateusz Żurawski, Mgr. Inż Jan Niekowal, Dr. Inż Paweł Rogaliński

ii. Członkowie zespołu: Maciej Jaroński, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko, Łukasz Handschuch

2. PRZEBIEG SPOTKANIA

Członkowie zespołu kolejno zaprezentowali swoje zadania za pomocą aplikacji discord.

- Maciej Jaroński : Task #35 Autoryzacja przy użyciu OAuth2.0(Frontend)
- Michał Juszczyk : Task#17 CRUD sprintu (Frontend)
- Łukasz Handschuch : Task#36 Operacje CRUD na zespole (Frontend)
- Nikita Stepanenko: Task#17 CRUD sprintu (Backend), Task#36 Operacje CRUD na zespole (Backend)
- Vladyslav Lutsenko: Task#35 Autoryzacja przy użyciu OAuth2.0(Backend)

Został utworzony sprint 5.Pan Mateusz z Panem Janem utworzyli nowe taski do zrobienia:

- Task #45 Dodawanie użytkownika do zespołu że zbioru użytkowników aplikacji
- Task #47 Uruchomienie aplikacji web na serwerze
- Task #33 Zaplanowanie spotkania w Sprincie
- Task #34 Wyświetlenie listy spotkań na dany dzień

3. WNIOSKI

Spotkanie przebiegło zgodnie z zaplanowaną procedurą. Zaplanowano szczegółowe podsumowanie pracy zespołu podczas sprintu (refiment) na wtorek 4 maja

Rys. 4.13: Raport 30.04

Raport Projektu Zespołowego

Data spotkania: 05.05.2020
Forma spotkania: Konferencja poprzez Discord

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal
   - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
   - Spotkanie miało formę weekly(podsumowanie wykonanego).
   - Wszyscy członkowie opowiedzieli o tym, co dało się wykonać do tej pory.
   - Zostało powiedziano o planach na następne spotkanie.
   - Została przeprowadzona retrospektywa.

3. **Podsumowanie**
   - Zostali omówione plany na przyszły tydzień.
   - Wynik retrospektywy:
     (+)
     *systematyczna praca (5)
     *praca zespołowa, dobra komunikacja zespołu w razie potrzeby, wszyscy nawzajem sobie pomagają(3)
     *przygotowana prezentacja, sprawnie zaprezentowane historyjki
     (-)
     *Sprint nie skończony (6) - analiza niewycenionych historyjek, spotkanie wewnętrzne? - lepsza wycena historyjek, empiryzm, - możliwość rozbicia historyjki na mniejsze.
     *Słabe rozplanowanie pracy, nieprzemyślane rozdzielenie tasków (5) - przemyślenie kolejności wykonywanych tasków, - zastanowienie się czy jakiś task/historyjka nie blokuje innego.
     *Połączenie BE i FE na samym końcu, były problemy(1) - mergowanie postępów pracy na branch develop.
     *na demo aplikacja nie była połączona(1) - aplikacja wrzucana na serwer przed każdym demo, - aktualizacja aplikacji na serwerze po ukończeniu zadania.
     *brak czasu

Rys. 4.14: Raport 05.05

Raport Projektu Zespołowego

Data spotkania: 12.05.2020
Forma spotkania: Konferencja poprzez Discord

1. **Obecni na spotkaniu:**
    - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal
    - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
    - Spotkanie miało formę weekly
    - Wszyscy członkowie opowiedzieli o tym, co dało się wykonać do tej pory.
    - Zostało powiedziano o planach na następne spotkanie.

3. **Podsumowanie**
    - Zostali omówione postępy w realizacji projektu.
    - Większość pracy została zrobiona, zostały tylko lekkie poprawki.
    - Wyjaśnione zostały kwestie sporne w implementacji

4. **Zadania do Wykonania**
    - Wszyscy członkowie projektu: dokończyć taski
    - Wszyscy członkowie projektu: przygotować się do DEMO
    - Maciek Jaroński: Załadować aplikację na serwer

Rys. 4.15: Raport 12.05

4.7. Sprint 6

Sprint trwał od 14.05.2020 do 28.05.2020
Zadania do Realizacji:

- Lista spotkań użytkowników
- Stworzenie dashboardu
- Usprawnienie komponentu tworzenia zespołu
- Uruchomienie aplikacji na serwerze
- Stworzenie landingpage dla projektu
- Usprawnienie procesu logowania

Raport Projektu Zespołowego

Data spotkania: 14.05.2020
Forma spotkania: Demo, konferencja poprzez Discord

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal
   - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**

Członkowie zespołu kolejno zaprezentowali swoje zadania za pomocą aplikacji discord:

- Michał Juszczyk: Task#33 Zaplanowanie spotkania w zespole (frontend)
- Łukasz Handschuch: Task#45 Dodawanie użytkownika do zespołu ze zbioru użytkowników aplikacji (frontend)
- Mcajiej Jaronicki: Task#34 Lista spotkań użytkownika (frontend)
- Nikita Stepanenko: Task#33 Zaplanowanie spotkania w zespole (backend), Task#34 Lista spotkań użytkownika (backend)
- Vladyslav Lutsenko: Task#45 Dodawanie użytkownika do zespołu ze zbioru użytkowników aplikacji (backend)

Został utworzony sprint 6. Pan Mateusz z Panem Janem utworzyli nowe taski do zrobienia:

- Task#34 Dodawanie użytkownika do zespołu ze zbioru użytkowników aplikacji (przeniesiony z poprzedniego sprintu)
- Task#54 Stworzenie dashboardu
- Task#56 Usprawnienie komponentu tworzenia zespołu
- Task#47 Uruchomienie aplikacji web na serwerze (przeniesiony z poprzedniego sprintu)
- Task#53 Landing page dla projektu
- Task#55 Usprawnienie procesu logowania
- Task#57 Usprawnienie w tworzeniu spotkania

3. **Zadania do Wykonania**
   - Nikita Stepanenko - Task#34 (backend), Task#47
   - Vladyslav Lutsenko - Task#54(backend)
   - Michał Juszczyk - Task#53 (frontend), Task#54 (frontend)
   - Łukasz Hadschuh - Task#55 (frontend), Task#56 (frontend)
   - Maciej Jaroński - Task#34 (frontend)

4. **Podsumowanie**

Spotkanie przebiegło zgodnie z zaplanowaną procedurą.
Zaplanowanie spotkanie w formie Weekly i Retro na 19.05.2020
Zespół przedzielił między sobą utworzone taski na wewnętrznym spotkaniu.

Rys. 4.16: Raport 14.05

Raport Projektu Zespołowego

Data spotkania: 19.05.2020
Miejsce spotkania: serwis rozmów głosowych «Discord»

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal
   - Członkowie grupy: Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
   - Spotkanie miało formę weekly(podsumowanie wykonanego).
   - Wszyscy opowiedzieli o tym co dało się do tej pory wykonać.
   - Została przeprowadzona Retrospektywa. Wynik:

(+)
-Dobra komunikacja, Pomoc/Komunikacja (2)
- połączenie aplikacji w całość (1)
- generalnie dobry postęp prac (1)
(-)
- Nie wszystkie taski wykonane ,niektóre detale historyjek do poprawy, drobne demo 'remarki' (6)
  • lepiej rozplanować harmonogram prac (taski są większe niż wyglądają)
  • zaplanować bufor czasowy
- Problemy z serwerem (2)
  • W przypadku problemu wysłać dokładne informacje doktorowi(dzien i godzina)
- aplikacja nie postawiona na serwerze (2)
- weryfikacja wymagań historyjki (1)
  • zaglądanie do wymagań historyjki
  • prośba o doprecyzowanie historyjki przed wyceną i na refinemencie

Rys. 4.17: Raport 19.05

21.05.2020

NOTATKA Z PROJEKTU ZESPOŁOWEGO

1. NAGŁÓWEK 1

Dnia 2105.2020 odbyło się spotkanie typu Daily Scrum. Przedstawiono na nim postępy pracy bieżącego sprintu.

A. Obecni na spotkaniu

i. Opiekunowie projektu: Mgr. Inż Mateusz Żurawski, Mgr. Inż Jan Niekowal, Dr. Inż Paweł Rogaliński

ii. Członkowie zespołu: Maciej Jaroński, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko, Łukasz Handschuch

2. PRZEBIEG SPOTKANIA

- Spotkanie miało formę Daily Scrum
- Michał Juszczyk przyjął funkcję Scrum Mastera podczas spotkania
- Członkowie zespołu kolejno zaprezentowali swoje zadania za pomocą aplikacji discord.
- Każdy z członków grupy opowiedział o swoich postępach w pracy nad projektem
- Zostały przypomniane oraz doprecyzowano zadania
- Omówiono sposoby rozwiązywania problemów

3. WNIOSKI

Spotkanie przebiegło zgodnie z zaplanowaną procedurą. Udało się wystawić aplikacje na serwer za pomocą docker-compose.

4. ZADANIA DO WYKONANIA

- Łukasz Handschuh – Usprawnienie komponentu tworzenia zespołu (Frontend), Usprawnienie procesu logowania.
- Nikita Stepanenko – Wystawienie aplikacji na serwerze
- Vladyslav Lutsenko – Logout (backend)
- Michał Juszczyk – Landing page dla projektu, Stworzenie Dashboardu

Rys. 4.18: Raport 21.05

Raport Projektu Zespołowego

Data spotkania: 28.05.2020
Forma spotkania: Konferencja poprzez Discord

1. **Obecni na spotkaniu:**
    - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal
    - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
    - Spotkanie miało formę demo
    - Wszyscy członkowie prezentowali rezultaty wykonanej pracy w sprincie.
    - Kolejne historyjki zostały wycenione.
    - Zaplanowany został następny sprint.

3. **Podsumowanie**
    - Wszystkie historyjki zostały zaakceptowane
    - Stworzony został kolejny sprint – tym razem tygodniowy
    - Dodane zostały historyjki do sprintu o łącznym koszcie analogicznym do długości sprintu

4. **Zadania do Wykonania**
    - Cały zespół musi zrealizować historyjki
    - Uruchomić aplikację Angularową jako PWA i tak uruchomić na serwerze.
    - Należy zarejestrować się na konferencję i zaplanować scenariusz

Rys. 4.19: Raport 28.05

4.8. Sprint 7

Sprint trwał od 28.05.2020 do 04.06.2020
Zadania do Realizacji:

- Rejestracja na konferencje projektów zespołowych
- Scenariusz prezentacji
- Nadawanie roli przez Scrum Mastera
- Aplakacja uruchomiona na serwerze jako PWA
- Notyfikacje o zaplanowanych wydarzeniach

Raport Projektu Zespołowego

Data spotkania: 03.06.2020
Forma spotkania: Weekly, Retrospectywa, konferencja poprzez Discord

1. **Obecni na spotkaniu:**
   - Opiekunowie projektu: Dr. Inż. Paweł Rogaliński, pracownik firmy Comarch Jan Niekowal
   - Członkowie grupy: Maciej Jaroński, Łukasz Handschuch, Michał Juszczyk, Stepanenko Nikita, Vladyslav Lutsenko.

2. **Przebieg Spotkania:**
   - podczas Weekly członkowie zespołu powiedzieli co na dzień dzisiejszy jest zrobiono i co zaplanowano do zrobienia w ramach sprintu.
   - Została przeprowadzona Retrospektyw, podczas której były przedstawione problemy napotkane podczas poprzedniego sprintu w celu ulepszenia pracy oraz rzeczy, którymi zespół został zadowolony. Wynik:

```
(+)
  Wszystko zostało zrobione na czas (4)
  Brak uwag do dema (2)
  Dobra komunikacja (1)
  dużo poprawek aplikacji (1)
  Prezentacja na serwerze (1)
  demo zgodne z wymaganiami historyjki
  Idealny sprint
  Pan Doktor brał udział w demo pośrednio
  dobra komunikacja
(-)
  Aplikacja nie działała jako PWA (8)
      ~ Utworzono historyjkę
  Nie było PWA na GitLab (4)
      ~ Częstsze wrzucanie zmian do repozytorium
      ~ Pobieranie bieżących wersji kodu
  Brak przesłanej informacji emailem o problemach z serwerem
      ~ Raportowanie Doktorowi o preblemach z serwerem (godzina i
  data oraz rodzaj problemu) - Wysłać maila
  Problemy podczas prezentowania dema
      ~ Wstępny plan demonstracji
  Serwer nie działał sprawnie podczas prezentacji
```

3. **Podsumowanie**
   Podczas spotkania było przeprowadzone Weekly oraz Retrospektywa

Rys. 4.20: Raport 03.06

Rozdział 5

Podsumowanie

5.1. Podsumowanie

Jako zespół otrzymaliśmy niełatwe zadanie. Musieliśmy nauczyć się całkowicie nowych technologii i stworzyć funkcjonalną aplikację. Dużą pomocą była wiedza dwóch pracowników firmy Comarch, których pomagali nam przez cały czas i pokazywali nam jak pracować w metodyce Scrum. Również Pan Doktor pomógł nam zorganizować pomoc ze strony politechniki np. serwer oraz pomagał nam w przypadku problemów, gdy był w stanie.

Pomimo wszystkich trudności udało się nam pracować regularnie w Scrum'ie co sprawiało, że wraz z każdym kolejnym sprintem poszerzała się funkcjonalność naszej aplikacji.

Ostateczna wersja okazała się wyjątkowo funkcjonalna. Możemy zarejestrować się i zalogować, a na podstawie tej weryfikacji użytkownik dostaje dostęp tyko do swoich informacji. Użytkownik może tworzyć własne zespoły oraz może być dodany do innych zespołów. Jeżeli ma uprawnienia może tworzyć nowe sprinty i spotkania, a jeżeli nie może zarządzać i organizować swoje spotkania.

Jesteśmy zadowoleni również z estetyki aplikacji. Całość wygląda nienagannie, jest prosta i intuicyjna. Podobna sytuacja dotyczy strony serwera. Mimo że ona nie posiada wersji graficznej dostępnej dla użytkownika, wszystkie zapytania do serwera odbywają się według schematu, który jest prosty i możliwy do wykorzystania w innych aplikacjach.

Gotowa aplikacja jest w pełni funkcjonalna. Jest możliwość wykorzystania tej aplikacji w praktyce po zakupienia odpowiedniego serwera. Instalacja byłaby prosta, ponieważ wszystko jest przygotowane w postaci obrazów dockerowych gotowych do wykorzystania po podłączeniu w odpowiedni sposób.

Całość wykonanej pracy oceniamy pozytywnie. Nauczyliśmy się wielu nowych umiejętności i pracy z technologiami, które pomogą nam w pisaniu pracy inżynierskiej i w pracy.

Aplikacja pracuje i będzie pracować dopóki serwer będzie dla nas dostępny i będzie prawidłowo pracował.

Bibliografia

[1] https://angular.io/docs - dokumentacja framework-a Angular,

[2] https://docs.oracle.com/javase/8/docs/ - dokumentacja języka Java8,

[3] https://docs.spring.io/spring/docs/ - dokumentacja framework-a Spring,

[4] https://docs.mongodb.com/ - dokumentacja MongoDB.