Przedmiot: Systemy zarządzania treścią
Projekt –

Piotr Góreczny

Nr indeksu 141224

e-mail: piotr.goreczny@student.put.poznan.pl

Spis treści

Charakterystyka ogólna projektu

Strona internetowa dla zespołu muzycznego z systemem zarządzania treścią pozwalającym na modyfikowanie zawartości. Edycja strony jest dostępna dla administratorów poprzez zabezpieczony hasłem panel administracyjny.
Strona ma zawierać aktualności, dyskografię zespołu z osobnymi podstronami dla każdego albumu oraz podstrony z informacjami kontaktowymi o zespole.
Strona jest oparta na wzorze Free Music Band Responsive Website Design ze strony templatemonster.com.

Wymagania

Wymagania funkcjonalne

Użytkownik może

- wyświetlić 3 najbardziej aktualne wiadomości na stronie głównej,
- Wyświetlić aktualne wydarzenia na stronie głównej,
- przejść do strony aktualności,
- przeglądać wiadomości na stronie aktualności,
- wyświetlić listę albumów,
- wyświetlić stronę konkretnego albumu,
- wyświetlić stronę z informacjami o zespole,
- wyświetlić podstronę
- przejść do mediów społecznościowych zespołu.

Administrator może

- wgrać zdjęcie do górnego banneru,
- dodawać/usuwać/edytować wiadomości do strony aktualności,
- dodawać/usuwać/edytować wydarzenia,
- dodawać/usuwać/edytować albumy,
  Albumy zawierają:
  - Nazwę
  - Zdjęcie
  - Tekst
  - Datę premiery
- Zarządzać(dodawać, usuwać, zarządzać kolejnością i edytować) podstronami zawierającymi tylko kolumnę tekstu.
- Edytować podstrony,
- Edytować oraz pokazywać/ukrywać linki do mediów społecznościowych.

Wymagania pozafunkcjonalne

- Czas ładowania strony głównej musi być nie dłuższy niż 1 sekunda.
- Czas ładowania strony albumu musi być nie dłuższy niż 1 sekunda.
- Strona powinna być zapopulowana 2 albumami i 5 wiadomościami
- Dostęp do panelu administracyjnego powinien być zabezpieczony loginem i hasłem.

Architektura systemu i narzędzia

Narzędzia

- Website template
- IntelliJ
- GitHub
- Google Chrome
- PgAdmin

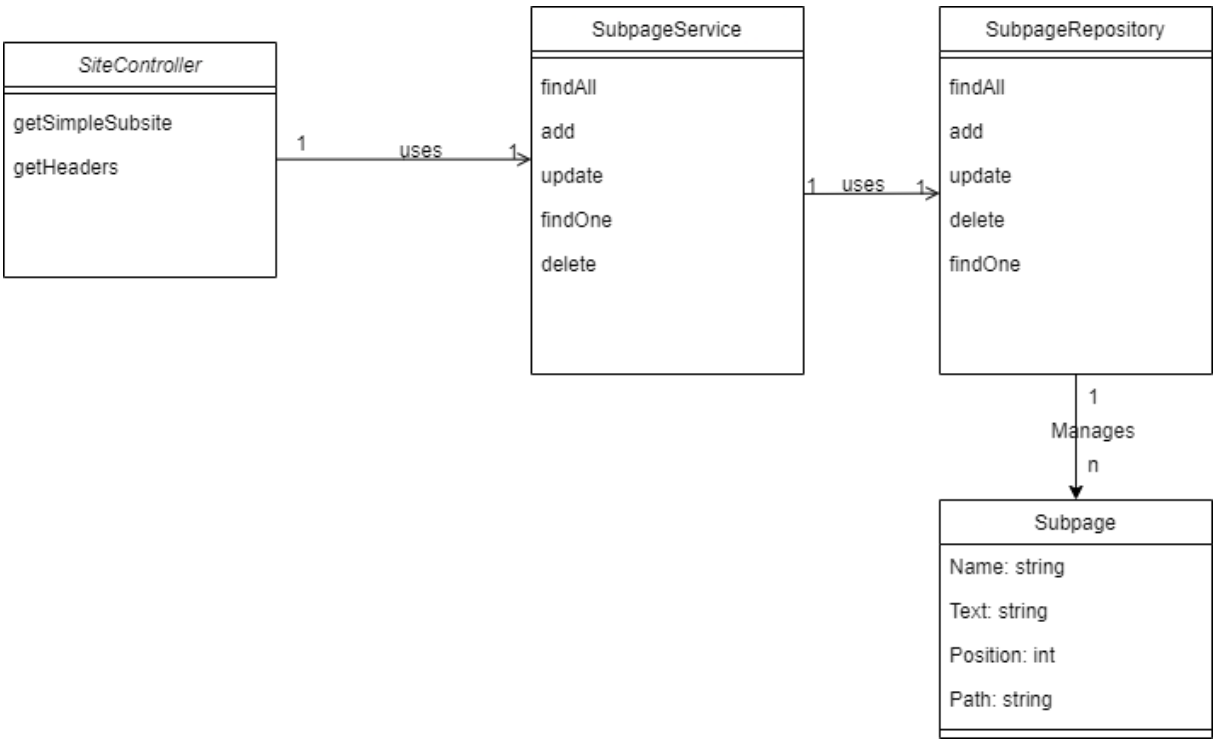
Technologia

- Java 11
- Spring Boot
- Spring Security
- PostgreSQL
- ThymeLeaf

Diagramy UML

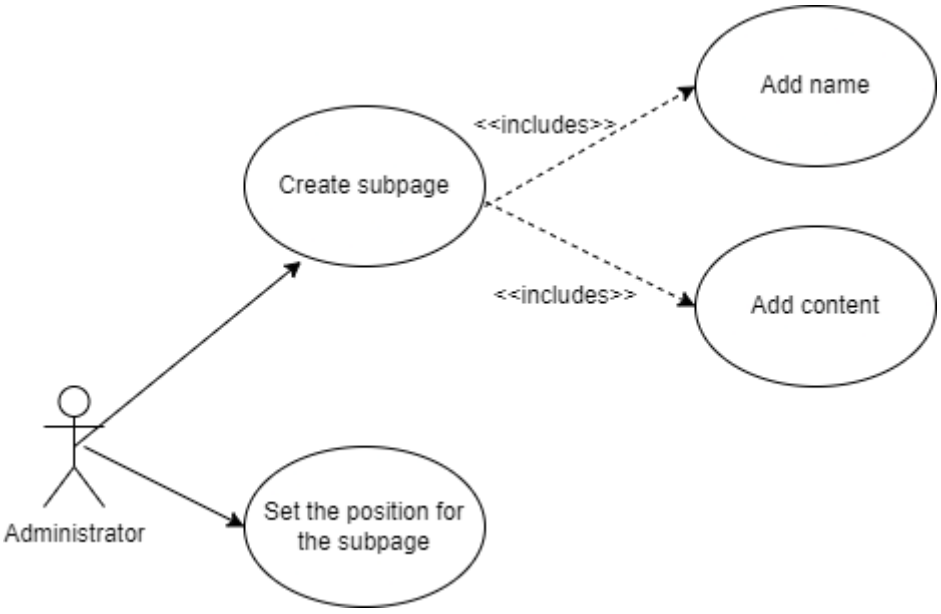
Diagramy UML prezentujące poszczególne aspekty działania projektowanego serwisu.

Diagram Klas



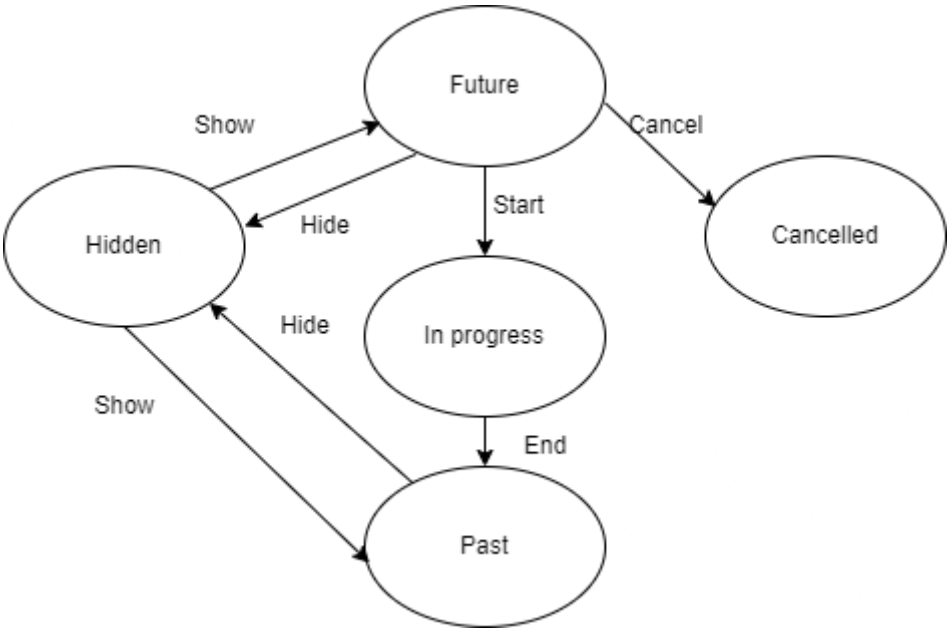
Rys. 1 – Diagram klas dla podstrony

Diagram przypadków użycia



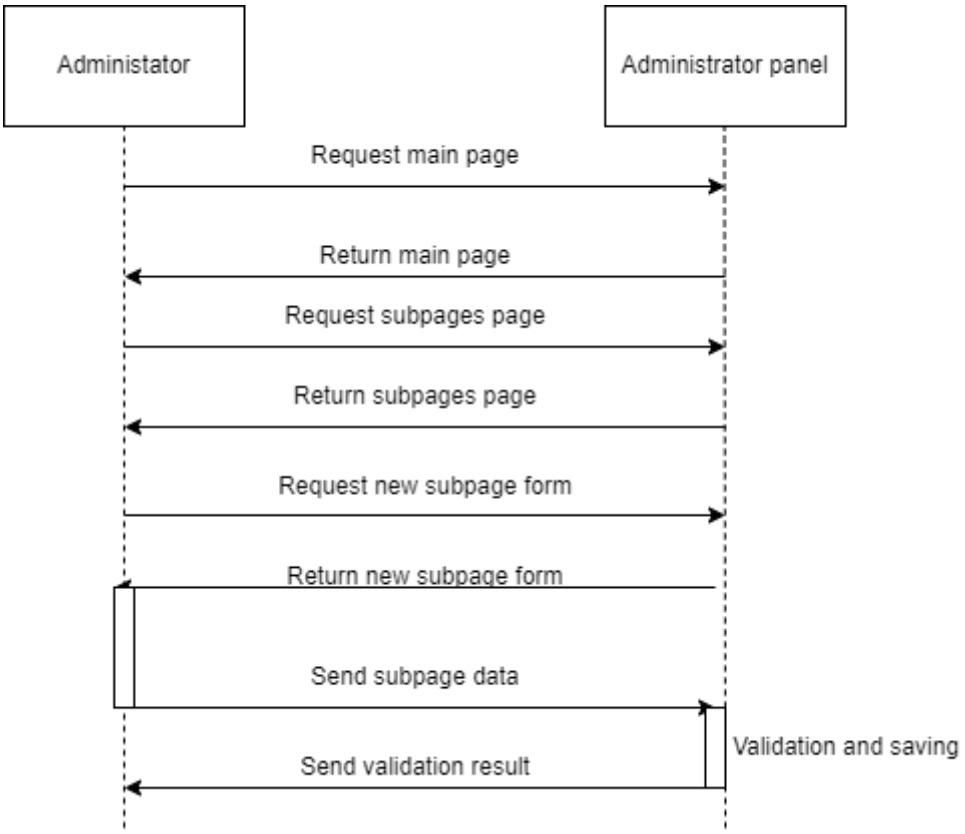
Rys. 2 – Diagram przypadku użycia dla zarządzania podstroną

Diagram stanów



Rys. 3 – Diagram stanów dla statusu wydarzeń

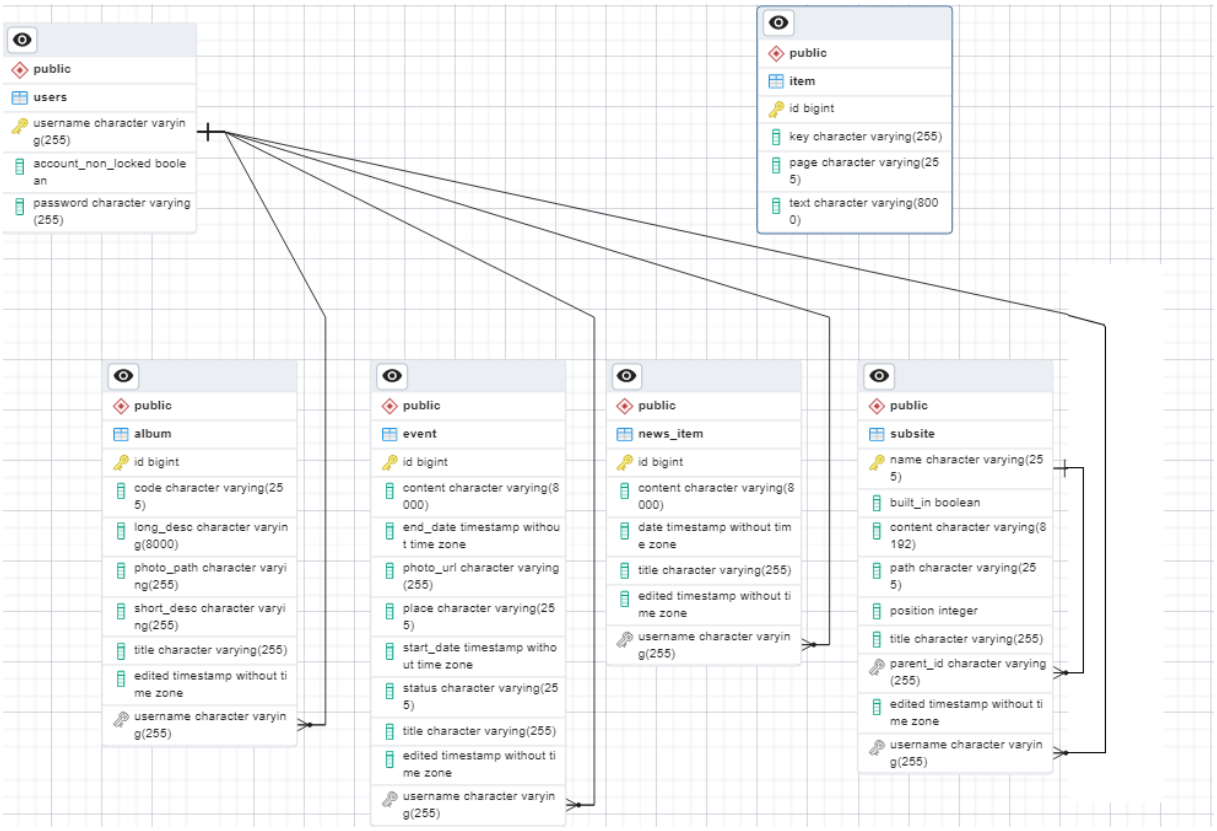
Diagram przebiegu



Rys. 4 – Diagram przebiegów dla dodawania podstrony

Model relacyjny

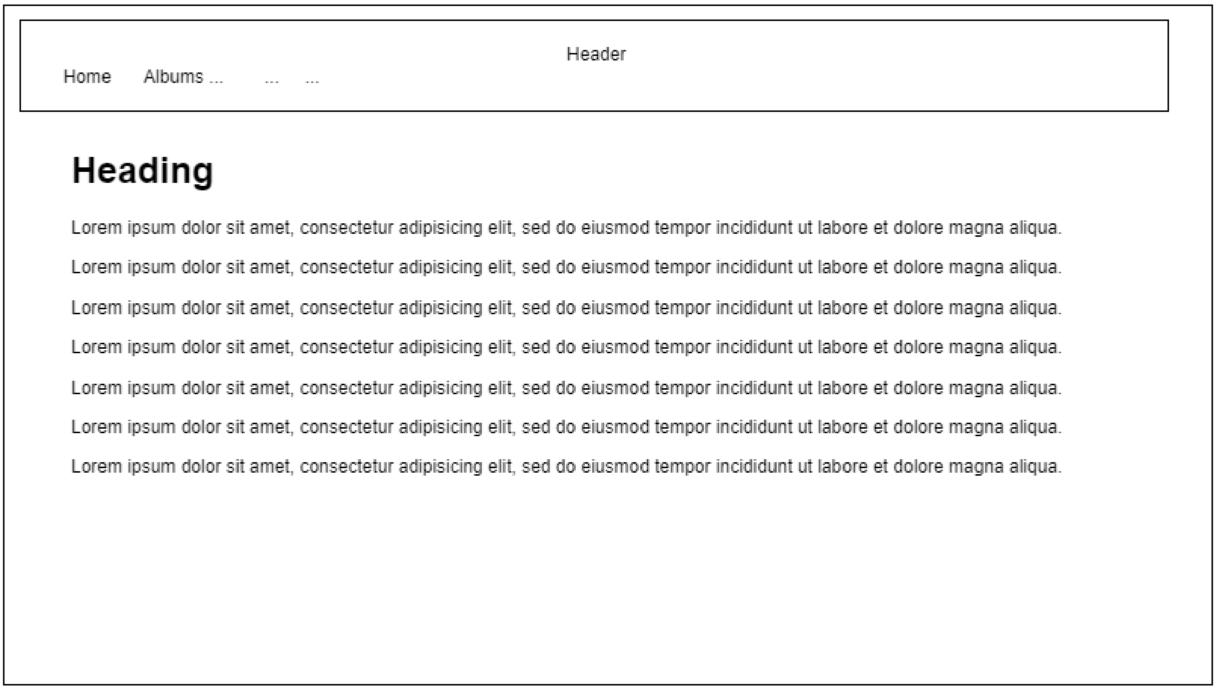
Baza danych jest stworzona na serwerze PostgreSQL i zawiera tabele odpowiadające modelom z aplikacji.



Rys. 5 – diagram bazy danych

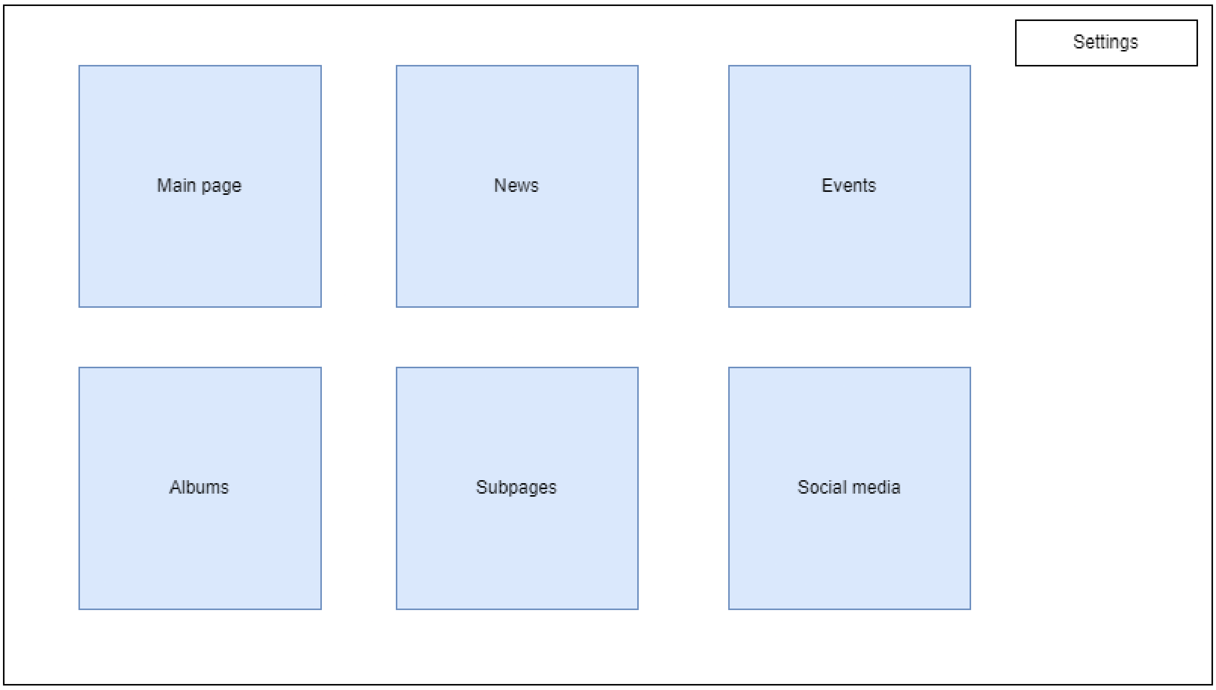
Interface graficzny

Projekt przykładowej podstrony



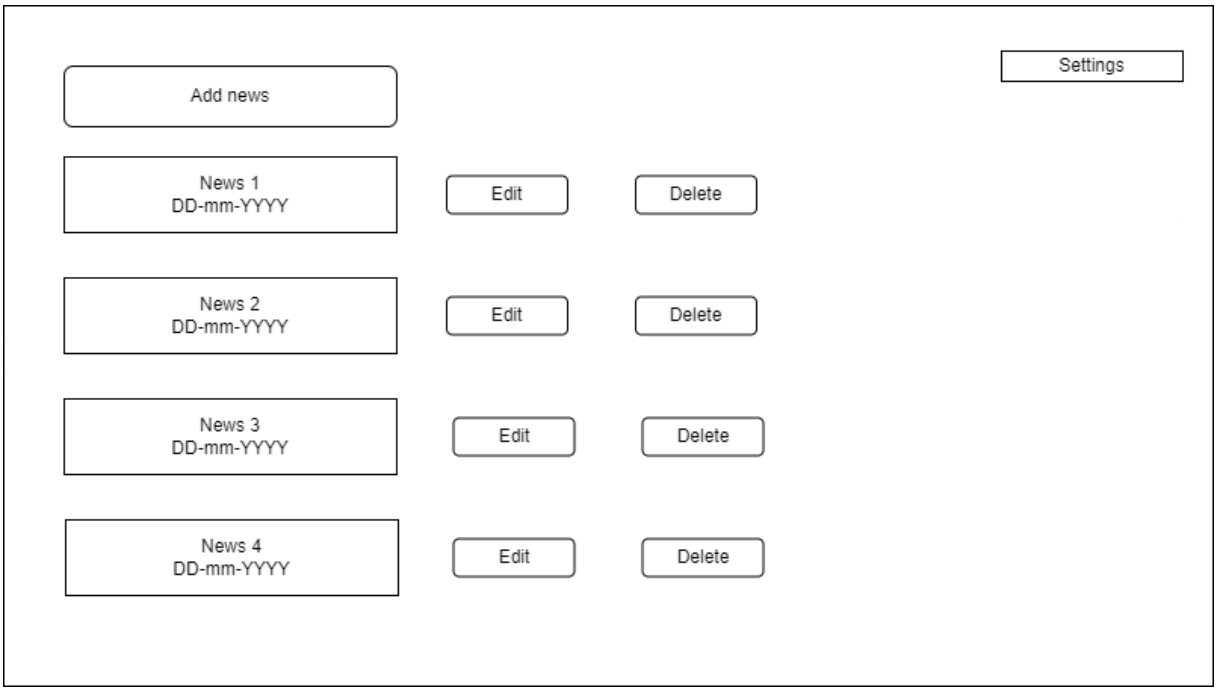
Rys. 6 - Projekt przykładowej podstrony

Projekt ekranu głównego interface'u administratora



Rys. 7 - Projekt ekranu głównego interface'u administratora

Projekt ekranu zarządzania newsami



Rys. 8 - Projekt ekranu zarządzania newsami

```
@GetMapping
public String getMainPage(Model model) {
    List<String> albums = albumService.getAlbums()
            .stream()
            .map(album -> album.getTitle())
            .collect(Collectors.toList());
    List<Subsite> subsites = subsitesService.getHeadSubsites();
    Map<String, String> items =
itemService.getItemsByPage(Arrays.asList("main", "common"));
    List<Event> events = eventService.getEventsVisible(2);
    List<NewsItem> news = newsService.getNews(3);
    Map<String, String> socialMedia =
itemService.getItemsByPage("socialMedia");

    model.addAttribute("albums", albums);
    model.addAttribute("subsites", subsites);
    model.addAttribute("items", items);
    model.addAttribute("events", events);
    model.addAttribute("news", news);
    model.addAttribute("socialMedia", socialMedia);
    return "website/index/index";
}
```

Fragment kodu 1 – Przygotowanie danych dla głównej strony aplikacji

```
@Entity
public class Subsite {
    @Id
    String name;
    @Transient
    private Long parentId;
    @Column
    String title;
    @Column
    String content;
    @Column
    int position;
    @Column
    String path;
    @Column
    boolean builtIn;
    @ManyToOne(fetch = FetchType.LAZY, optional=true)
    @JoinColumn(name="parent_id")
    Subsite parent;
    @OneToMany(mappedBy="parent", fetch = FetchType.LAZY, cascade =
            CascadeType.ALL, orphanRemoval=true)
    List<Subsite> children;
    @Column
    Date edited;
    @ManyToOne
    @JoinColumn(name="username")
    User editedBy;
```

Fragment kodu 2 – klasa reprezentująca podstronę

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception
{
    http
            .crlf().disable()
            .authorizeRequests()
            .antMatchers("/admin/**").authenticated()
            .anyRequest().permitAll()
            .and()
            .formLogin()
            .permitAll()
            .and()
            .logout() .invalidateHttpSession(true)
            .clearAuthentication(true) .permitAll();
    return http.build();
}
```

Fragment kodu 3 – process obsługi zapytania i uwierzytelnienia

Analiza bezpieczeństwa

1. Strona zespołu jest publicznie dostępna i nie przewiduje możliwości wprowadzania danych, co sprawia, że ta część serwisu nie ma za dużo zagrożeń.
2. Dostęp do panelu administracyjnego jest zabezpieczony hasłem.
3. Po stronie serwera przechowywany jest jedynie zasolony hash hasła, co chroni przed wyciekiem hasła.
4. Uwierzytelnianie jest obsługiwane przy użyciu biblioteki Spring Security.
5. Zapytania są zabezpieczone przed CRLF injection, input jest transformowany do obiektów, a nie zapisywany w formie surowej.
6. Wszystkie dane są wprowadzane przy użyciu mechanizmów JPA, serwis nie zapisuje surowych zawartości pól wprowadzonych przez użytkownika, ale tworzy z nich obiekty, które następnie przekazuje do repository do zapisu , co chroni przed SQL injection.
7. W przypadku próby odwołania się do obiektu(albumu, podstrony etc) o błędnym identyfikatorze w panelu administracyjnym, zostanie to wychwycone, a użytkownikowi zaproponowane zostanie utworzenie nowego obiektu.

Podsumowanie

Java nie jest typowym wyborem dla systemów CMS, ale w ramach tego projektu sprawdza się dobrze. Do zalet należy klarowność i łatwość utrzymania kodu, dobre mechanizmy bezpieczeństwa i efektywna współpraca z bazą danych. Do wad należał większy niż typowy narzut pracy na początku w celu skonfigurowania projektu i narzucone konwencje strukturalne.

Ponadto domyślnie Spring ładuje static content(templatki stron, style i pliki graficzne) przy uruchomieniu i nie zakłada ich modyfikowania w trakcie działania programu. Obejście tego problemu pochłonęło sporo czasu i wymagało skonfigurowania resource handlera i przechowywania plików graficznych poza ścieżką roboczą programu, aby umożliwić administratorowi dodawanie i zmianę grafik.

Stan końcowy

Wszystkie założone funkcjonalności są dostępne dla użytkownika a administrator ma możliwość zarządzania założonymi obiektami(albumy, kolejność podstron i możliwość dodawania nowych, wiadomościami, wydarzeniami i linkami do profili w portalach społecznościowych.)