

Projekt semestralny - założenia funkcjonalne

Wiktor Urban

May 15, 2023

Abstract

Projekt na przedmiot Zaawansowane Techniki Internetowe.

1 Wymagania dotyczące projektu

Zaliczenie przedmiotu następuje po oddaniu projektu zrealizowanego w czasie trwania semestru. Tematem projektu jest aplikacja klient serwer zrealizowana zgodnie z wzorcem RESTful (Java API for RESTful Web Services JAX-RS), z wykorzystaniem technologii WebSocket (Java API for WebSocket) lub z interfejsem GraphQL. Część serwerowa wykorzystuje serwer bazy danych (relacyjny bądź NoSQL). Klient może być zrealizowany z wykorzystaniem serwisu WWW i języka Javascript lub może to być aplikacja mobilna. Temat projektu należy uzgodnić z prowadzącym laboratorium. Zaliczenie projektu następuje po zrealizowaniu następujących elementów:

- serwer - aplikacja serwerowa uruchomiona w ramach chmur: IBM Cloud, Microsoft Azure, Google Cloud, AWS Cloud lub z wykorzystaniem konteneryzacji;

- serwer - realizacja w technologii platformy Jakarta EE lub dodatkowo z wykorzystaniem technologii Spring i technologii Spring Boot;

- serwer - dostęp do danych zrealizowany w oparciu o odwzorowanie obiektowo-relacyjne JPA lub Spring Data;

- serwer - baza danych w rozwiązaniu chmurowym DBaaS;

- serwer - obsługa zdarzeń z wykorzystaniem programowania aspektowego - język AspectJ lub Spring AOP;

- klient - opracowany z wykorzystaniem serwisu WWW i języka JavaScript, aplikacja typu SPA lub jako aplikacja mobilna;

- klient - w ramach aplikacji można wykorzystać odpowiednie frameworki (np. AngularJS, React, Vue.js czy Cordova);

- dokumentacja - opis funkcjonalności opracowanej aplikacji zarówno części serwerowej jak i klienta;

- dokumentacja - dołączone odpowiednie diagramy UML dla części serwerowej i klienta;

- dokumentacja - prezentacja wybranych testów jednostkowych i wdrożeniowych dla części serwerowej i klienta;

- dokumentacja - informacja uruchomieniowa (wdrożeniowa);

- dokumentacja - kod aplikacji należy udokumentować przy pomocy JavaDoc (w przypadku aplikacji mobilnej również);

- dokumentacja - podstawowy podręcznik użytkownika.

2 O projekcie

Projekt, który zdecydowałem się przygotować jest aplikacją webową pozwalającą na dostęp do danych z bazy danych oraz na modyfikację tych danych. Problemem jaki aplikacja ma rozwiązać to sprawdzenie historii danego rekordu (na przykład lokacji) z dokładnością do timestampu zmiany pojedynczego pola.

Funkcjonalność edycji rekordu pozwala na edycję pojedynczych pól co jest odzwierciedlone w schemacie bazy danych. Dzięki temu możemy łatwo zobaczyć historię zmian (obrazek 8) jakim został poddany dany rekord. Ponadto użytkownik może "cofnąć się w czasie" i zobaczyć jaki był stan bazy o wybranym dniu oraz czasie. Można sprawdzić jakie rekordy zostały zmienione w wybranym przedziale czasowym. Te funkcjonalności są według mnie pomocne gdy ktoś chciałby prześledzić co działo się z danym rekordem.

3 Schemat bazy danych

Schemat blokowy został umieszczony na obrazku 1. Dany rekord zamiast tworzyć pojedynczej tabeli został rozbity w sposób taki aby każda cecha rekordu była w osobnej tabeli razem z timestampem zmian. Dzięki temu podejściu jesteśmy całkowicie śledzić zmiany na poziomie pojedynczych cech. Dany rekord zawiera się z połączonych relacją 1 - n lub n - m tabel. Przykładowo rekord Location składa się z tabel: location, location_is_active, location_street_address itd.

Na przykładzie tabeli location_street_address opiszę logikę tego schematu. Każda z tabel, która definiuje pole rekordu, składa się z czterech pól:

- id pola w tabeli, przykładowo location_street_address_id. Jest to sztuczne pole istniejące tylko po to aby tabela miała primary key. Wymagało tego narzędzie do tworzenia schematu ERD - SQL Power Architect.
- id rekordu do którego dana cecha jest przypisana, dla wyżej wymienionej tabeli jest to location_id.
- timestamp oznaczający kiedy dany rekord pola został dodany do tabeli pola.
- faktyczna wartość pola, dla tabeli powyżej jest to street_address.

W tym podejściu dane nigdy nie są usuwane lub nadpisywane. Zamiast tego przy edycji danego pola Lokacji, dodawany jest nowy rekord w odpowiedniej tabeli. Jeżeli chcemy deaktywować Lokację dodajemy nowy wpis w tabeli location_is_active z nowym timestampem i wartością is_active ustawioną na false.

Jest to podejście alternatywne do przechowywania rekordu Location w jednej tabeli z wszystkimi polami tego rekordu. Zaletą mojego rozwiązania jest zmniejszenie ilości pamięci użytej na przechowywanie danych, ponieważ przy aktualizacji pojedynczego pola Lokacji zostaje dodany wpis do tabeli związanej z danym polem - zawierającej w większości przypadków 4 kolumny. W podejściu klasycznym jeżeli Lokacja zawierałaby 10 pól, zostałby dodany wpis zawierający 10 pól pomimo że zmienione zostało tylko jedno z nich. Kolejną zaletą mojego rozwiązania jest stosunkowo łatwe rozproszenie bazy danych na wiele instancji. W skrajnym przypadku tabela powiązana z pojedynczym polem mogłaby być w osobnej instancji. Wadą mojego rozwiązania jest większe skomplikowanie tabel oraz konieczność użycia funkcji i procedur postgresql łączących tabele. Koszt wyszukiwania rekordów w bazie jest zmniejszony dzięki dodaniu indexów na polach location_id w tabelach zawierających to pole.

Przykładowy kod do funkcji za pomocą której pobieram listę obecnie aktualnych lokacji umieszczam poniżej. Analizując ten kod można łatwo zauważyć, że pola rekordu są składane z wartości z wielu tabel, gdzie każda z tabel odnosi się do pojedynczej cechy.

```
1 CREATE OR REPLACE FUNCTION zti_projekt2.get_current_locations()
2 RETURNS TABLE(location_id INTEGER, is_active boolean, street_address varchar,
3 city varchar, zipcode varchar, state varchar, country_code varchar,
4 country_code_id integer, is_country_code_active boolean, activity_name varchar,
5 activity_id integer, is_activity_active boolean, company_name varchar
6 ) AS $$
7 BEGIN
8 RETURN QUERY
9 SELECT zti_projekt2.location.location_id,
10 (select zti_projekt2.location_is_active.is_active
11 from zti_projekt2.location_is_active
12 where zti_projekt2.location.location_id = zti_projekt2.location_is_active.
13 location_id
14 order by timestamp desc
15 limit 1
16 ),
17 (select zti_projekt2.location_street_address.street_address
18 from zti_projekt2.location_street_address
19 where zti_projekt2.location.location_id = zti_projekt2.location_street_address.
20 location_id
21 order by timestamp desc
22 limit 1
23 ),
24 (select zti_projekt2.location_city.city
```

```

21 from zti_projekt2.location_city
22 where zti_projekt2.location.location_id = zti_projekt2.location_city.location_id
23 order by timestamp desc
24     limit 1
25 ),
26 (select zti_projekt2.location_zipcode.zipcode
27 from zti_projekt2.location_zipcode
28 where zti_projekt2.location.location_id = zti_projekt2.location_zipcode.location_id
29 order by timestamp desc
30     limit 1
31 ),
32 (select zti_projekt2.location_state.state
33 from zti_projekt2.location_state
34 where zti_projekt2.location.location_id = zti_projekt2.location_state.location_id
35 order by timestamp desc
36     limit 1
37 ),
38
39
40 (select zti_projekt2.country_code_code.country_code
41 from zti_projekt2.location_country_code
42 join zti_projekt2.country_code_code on zti_projekt2.country_code_code.
43     country_code_id = zti_projekt2.location_country_code.country_code_id
44 where zti_projekt2.location.location_id = zti_projekt2.location_country_code.
45     location_id
46 order by location_country_code.timestamp desc, country_code_code.timestamp desc
47     limit 1
48 ),
49
50 (select zti_projekt2.location_country_code.country_code_id
51 from zti_projekt2.location_country_code
52 where zti_projekt2.location.location_id = zti_projekt2.location_country_code.
53     location_id
54 order by location_country_code.timestamp desc
55     limit 1
56 ),
57
58 (select zti_projekt2.country_code_is_active.is_active
59 from zti_projekt2.location_country_code
60 join zti_projekt2.country_code_is_active on zti_projekt2.country_code_is_active.
61     country_code_id = zti_projekt2.location_country_code.country_code_id
62 where zti_projekt2.location.location_id = zti_projekt2.location_country_code.
63     location_id
64 order by location_country_code.timestamp desc, country_code_is_active.timestamp desc
65     limit 1
66 ),
67
68
69 (select zti_projekt2.activity_name.activity_name
70 from zti_projekt2.location_activity
71 join zti_projekt2.activity_name on zti_projekt2.activity_name.activity_id =
72     zti_projekt2.location_activity.activity_id
73 where zti_projekt2.location.location_id = zti_projekt2.location_activity.location_id
74 order by location_activity.timestamp desc, activity_name.timestamp desc
75     limit 1
76 ),
77
78 (select zti_projekt2.location_activity.activity_id
79 from zti_projekt2.location_activity
80 where zti_projekt2.location.location_id = zti_projekt2.location_activity.location_id
81 order by location_activity.timestamp desc
82     limit 1
83 ),
84
85 (select zti_projekt2.activity_is_active.is_active
86 from zti_projekt2.location_activity
87 join zti_projekt2.activity_is_active on zti_projekt2.activity_is_active.
88     activity_id = zti_projekt2.location_activity.activity_id
89 where zti_projekt2.location.location_id = zti_projekt2.location_activity.location_id
90 order by location_activity.timestamp desc, activity_is_active.timestamp desc
91     limit 1
92 ),

```

```

85     (select zti_projekt2.location_company_name.company_name
86 from zti_projekt2.location_company_name
87 where zti_projekt2.location.location_id = zti_projekt2.location_company_name.
      location_id
88 order by timestamp desc
89      limit 1
90     )
91 FROM zti_projekt2.location;
92
93 END;
94 $$ LANGUAGE plpgsql;
95
96
97
98
99 select  * from zti_projekt2.get_current_locations();

```

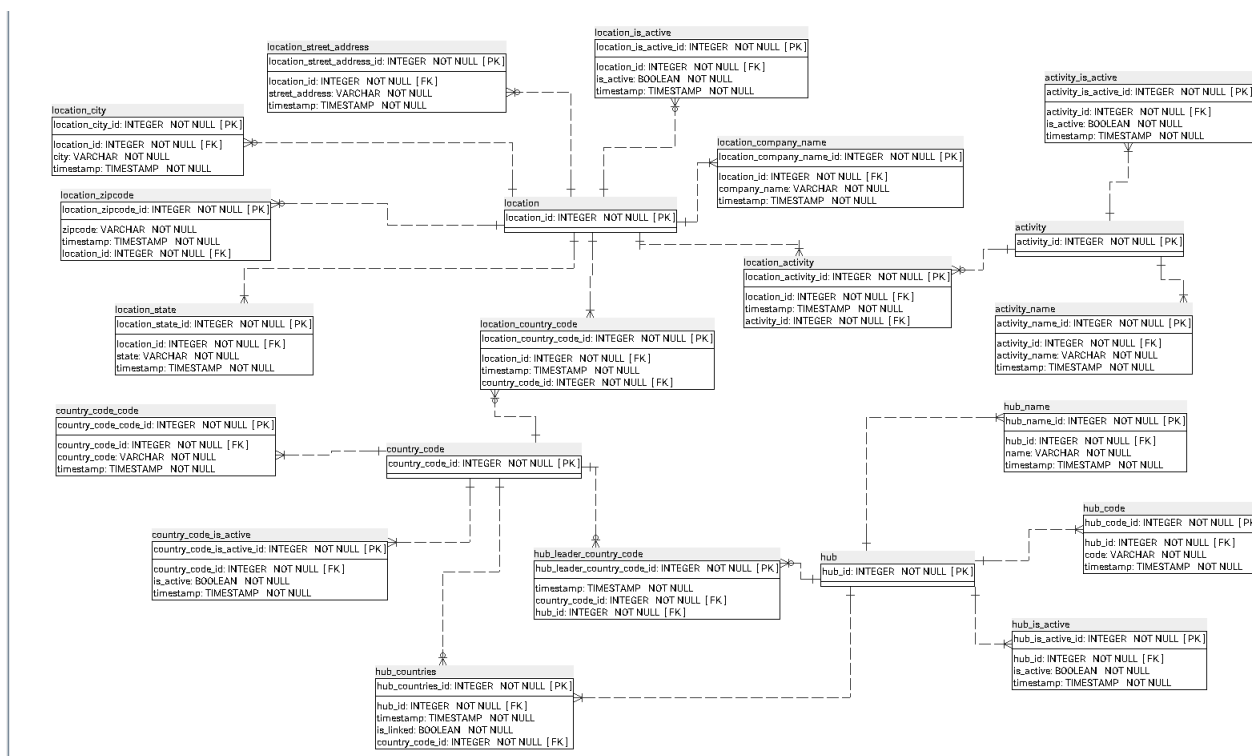


Figure 1: Schemat bazy danych

4 Założenia funkcjonalne

- możliwość zobaczenia listy aktualnych rekordów - done fig.2.
- możliwość sprawdzenia historii edycji danego rekordu (gdzie rekord to na przykład location) - done fig.8
- możliwość zobaczenia stanu listy rekordów aktualnych w wybranym danym dniu i czasie - done fig.6
- możliwość sprawdzenia ilości rekordów zmodyfikowanych pomiędzy dwoma datami oraz szczegółów zmian dla wybranego rekordu - done fig.7 fig.9
- możliwość dodania i edycji rekordów oraz deaktywacji i reaktywacji - done fig.3 oraz fig.4

5 Obecny postęp

Baza danych to postgresql. Backend został wykonany z użyciem Springa. Strona frontendowa została zaimplementowana przy pomocy frameworku Angular 15.

Dostęp do bazy danych jest poprzez JPA. Operacje na rekordach zostały zaimplementowane przy użyciu procedur oraz funkcji wewnątrz bazy danych postgresql.

Przykładowa metoda pobierająca listę obecnie aktualnych rekordów z bazy danych.

```

1
2 List<Location> getCurrentLocations(){
3     String query = "SELECT * FROM zti_projekt2.get_current_locations()";
4     return jdbcTemplate.query(query, (rs, rowNum) -> {
5         int locationId = rs.getInt("location_id");
6         boolean isActive = rs.getBoolean("is_active");
7         String streetAddress = rs.getString("street_address");
8         String city = rs.getString("city");
9         String zipcode = rs.getString("zipcode");
10        String state = rs.getString("state");

```

























Current Locations									
locationId	isActive	streetAddress	city	zipcode	state	countryCode	activityName	companyName	Actions
4	false	213 Main Ste	Krakowd	30-123	MA	ZN	sales	ABC Company	 
5	false	123 Main St	grzeszow	12345	CA	CH	sales	heheh corp	 
6	true	123 Main St	Krakow	30-123	MA	PL	r&d	ABC Company	 
7	true	dfdf	dddd	dfdf	ffefef	ZN	r&d	gthytit	 
8	true	123 Main St	Krakow	30-123	MA	FR	r&d	ABC Company	 
9	true	robr	england is my city	436534	hampshire	FR	r&d	dynamics	 

Figure 2: Widok listy lokacji dla stanu obecnego

Current Locations									
locationId	isActive	streetAddress	city	zipcode	state	countryCode	activityName	companyName	Actions
4	false	213 Main Ste	Krakowd	30-123	MA	ZN	sales	ABC Company	 
5	false	123 Main St	grzeszow	12345	CA	CH	sales	heheh corp	 
6	true	123 Main St	Krakow	30-123	MA	PL	r&d	ABC Company	 
7	true	dfdf	dddd	dfdf	ffefef	ZN	r&d	gthytit	 
8	true	123 Main St	Krakow	30-123	MA	FR	r&d	ABC Company	 
9	true	robr	england is my city	436534	hampshire	FR	r&d	dynamics	 

Update data.location 4

streetAddress	city	zipcode	state	companyName
213 Main Ste	Krakowd	30-123	MA	ABC Company

countryCode	activity
ZN	sales

Cancel Send Update

Figure 3: Dialog edycji lokacji.

```

11     String countryCode = rs.getString("country_code");
12     int countryCodeId = rs.getInt("country_code_id");
13     boolean countryCodeIsActive = rs.getBoolean("is_country_code_active");
14     String activityName = rs.getString("activity_name");
15     int activityId = rs.getInt("activity_id");
16     boolean activityIsActive = rs.getBoolean("is_activity_active");
17     String companyName = rs.getString("company_name");
18     return new Location(locationId, isActive, streetAddress, city, zipcode,
19         state, companyName,
20         new CountryCode(countryCodeId, countryCodeIsActive, countryCode),
21         new Activity(activityId, activityIsActive, activityName), null);
22     });

```

Obecnie zaimplementowane są trzy kontrolery: LocationController, ActivityController oraz CountryCodeController. ActivityController i CountryCodeController pozwalają na pobranie listy opcji do wyboru przy aktualizacji lub dodawania lokacji.

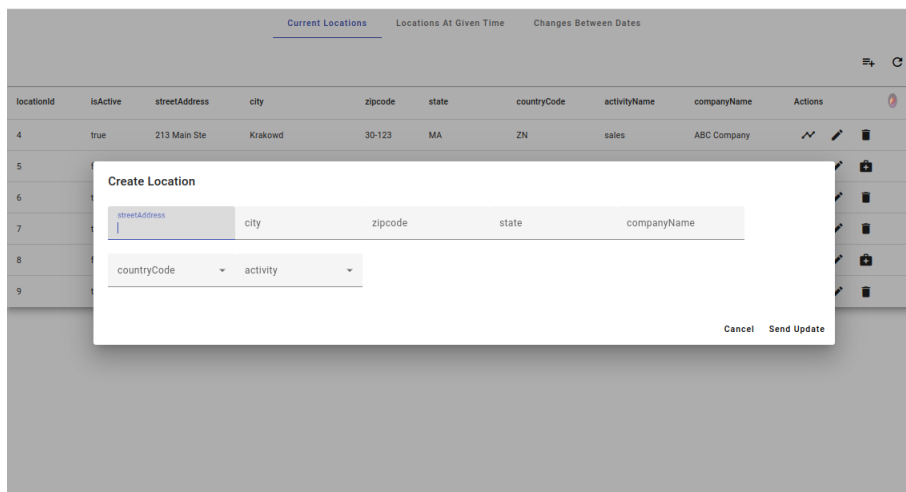


Figure 4: Dialog dodania lokacji.

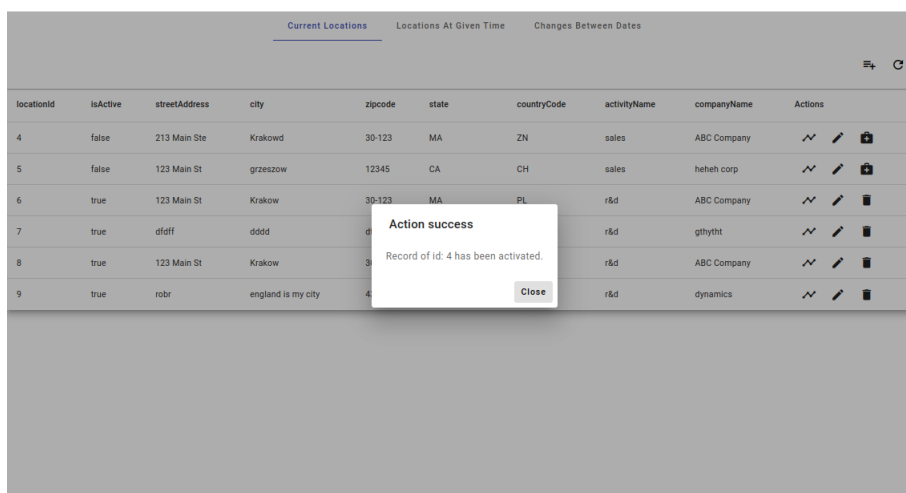


Figure 5: Dialog potwierdzający reaktywację rekordu.

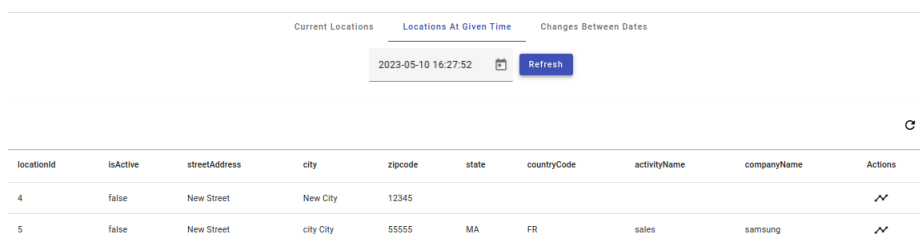


Figure 6: Widok listy lokacji dla stanu w wybranym dniu i godzinie.


Current Locations

Locations At Given Time

Changes Between Dates


start date

2023-05-13 19:27:52





end date

2023-05-14 19:27:52



Refresh

locationId	isActive	streetAddress	city	zipCode	state	countryCode	activityName	companyName	Actions
6	true	123 Main St	Krakow	30-123	MA	PL	r&d	ABC Company	
9	true	robr	england is my city	436534	hampshirefe	FR	r&d	dynamics	

Current Locations

Locations At Given Time

Changes Between Dates

start date

2023-05-13 19:27:52

end date

2023-05-14 19:27:52

Refresh

Timeline of location id: 6

timestamp	isActive	streetAddress	city	zipcode	state	countryCode	activityName	companyName
current	true	123 Main St	Krakow	30-123	MA	PL	r&d	ABC Company
2023-05-14 10:50:44.056275	true							
2023-05-11 10:24:59.255857	false					PL		
2023-05-11 10:24:52.060909	false							
2023-05-11 08:09:21.146734	true	123 Main St	Krakow	30-123	MA	FR	r&d	ABC Company

Close

Current Locations

Locations At Given Time

Changes Between Dates

start date

2023-05-13 19:27:52

end date

2023-05-14 19:27:52

Refresh

locationId	isActive	streetAddress	city	zipcode
6	true	123 Main St	Krakow	30-123
9	true	robr	england is my city	436534

activityName	companyName	Actions
r&d	ABC Company	
r&d	dynamics	