

Projekt BDBT

-

”Operator sieci komórkowej”

-

Część druga

Dana Betsina

Emilia Anczarska

Politechnika Warszawska, Instytut Telekomunikacji

26 stycznia 2024

Spis treści

1. Zakres i cel projektu	3
2. Opis wykorzystanej technologii	3
2.1. Środowisko - IntelliJ IDEA	3
2.2. Java	3
2.2.1. Java Spring	3
2.2.2. Spring Boot i Bootstrap	3
2.2.3. Spring Thymeleaf	4
2.2.4. Spring Test	4
2.2.5. Spring Security	4
2.3. Oracle JDBC	4
2.4. Webjars locator	5
2.5. Project Lombok	5
3. Działanie aplikacji	6
3.1. Pierwsza płaszczyzna Aplikacji	6
3.1.1. Intuicyjność	6
3.1.2. Przyrzystość	7
3.1.3. Pomysł	8
3.1.4. Jakość rozwiązania i obsługa błędów	11
3.2. Druga płaszczyzna Aplikacji	12
3.2.1. Perspektywa użytkownika	12
3.2.2. Perspektywa administratora	15

1. Zakres i cel projektu

Celem II części projektu było przygotowanie aplikacji współpracującej z wcześniej zaprojektowaną bazą danych z części pierwszej. Aplikacja nie musiała pokrywać w pełni funkcjonalności wcześniej zaprojektowanej bazy danych (jedynie część, kilka wybranych transakcji bazodanowych – kilka tabel). Należało zadbać o różnorodność wybranych tabel tzn., aby w aplikacji znajdowały się tabele, które są ze sobą powiązane (nie pojedyncze tabele). Aplikacja została napisana z użyciem technologii Java w architekturze wielowarstwowej.

2. Opis wykorzystanej technologii

2.1. Środowisko - IntelliJ IDEA

Do projektu wybrałyśmy środowisko pracy IntelliJ IDEA. To bardzo popularne zintegrowane środowisko programistyczne (IDE) opracowane przez JetBrains. Jest przeznaczony głównie dla języka Java, ale obsługuje także szeroką gamę języków programowania, w tym Kotlin, Groovy, Scala i inne.

2.2. Java

2.2.1. Java Spring

Architektura Spring stanowi zaawansowany przykład architektury szkieletowej umożliwiającej szybkie rozwijanie dowolnie złożonych aplikacji, niekoniecznie webowych. Może być ona wykorzystana na platformie internetowej do tworzenia dużych aplikacji Java EE. Konstrukcja szkieletu architektury Spring powoduje, że jest ona typem tzw. *lightweight framework*, w której występuje bardzo niewielki stopień zależności od interfejsów Spring API. Architektura Spring obejmuje wszystkie warstwy aplikacji i podsuwa rozwiązania, które mogą być stosowane zarówno w warstwie prezentacji, jak i w warstwie integracji i warstwie danych.

Architektura Spring obejmuje wszystkie warstwy aplikacji, proponując uniwersalne rozwiązania, które mogą być używane zarówno w warstwie prezentacji, jak i integrowania różnych elementów oraz zarządzania danymi. Opierając się głównie na interfejsach, rzadko korzysta z mechanizmu dziedziczenia, co stanowi świadomy wybór projektowy. Celem jest zwiększenie autonomii pomiędzy aplikacją a interfejsem programistycznym Spring oraz pomiędzy różnymi warstwami aplikacji. Kluczową ideą architektury Spring jest również podejście do obsługi wyjątków, umożliwiając niezależne i pojedyncze testowanie poszczególnych modułów aplikacji.

W skład szkieletu wchodzi:

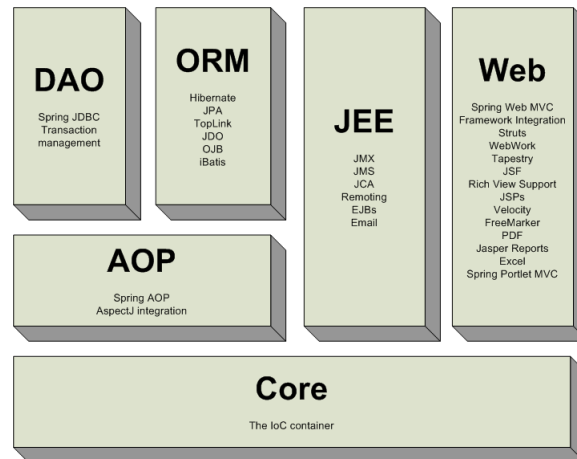
1. lekki kontener zarządzający komponentami JavaBean i prostymi klasami Java POJO w sposób deklaratywny na podstawie pliku konfiguracyjnego,
2. warstwa zarządzania transakcjami zgodna z wieloma standardami, takimi jak JTA, JDBC, JDO, Hibernate,
3. gotowy szablon obsługujący interfejs JDBC wraz z uproszczoną hierarchią wyjątków zgłaszanych przez źródła danych,
4. moduły w pełni integrujące się z popularnymi narzędziami zarządzania trwałością danych, takimi jak Hibernate, JDO, Toplink, czy iBATIS SQL Maps,
5. kontener umożliwiający pełne wykorzystanie programowania aspektowego,
6. kompletny szkielet aplikacyjny MVC z własnym kontrolerem i kontenerem obsługi JSP i JavaBeans, umożliwiający integrację z istniejącymi szablonami, np. Apache Struts, WebWork czy Tapestry,
7. narzędzia do łatwej integracji z technologiami szkieletowymi warstwy prezentacji, np. Velocity Templates i Struts Tiles.

2.2.2. Spring Boot i Bootstrap

Spring Boot jest rozszerzeniem frameworka Spring, które eliminuje standardowe konfiguracje wymagane do skonfigurowania aplikacji Spring. Pozwala na szybką budowę aplikacji w oparciu o przygotowaną kolekcję szablonów startowych (starter templates).

Tworząc aplikację webową z wykorzystaniem Spring MVC, wystarczy skorzystać z odpowiedniej zależności, która reprezentuje gotowy szablon do budowy aplikacji webowych. Poprzez używanie tego predefiniowanego szablonu, wszystkie niezbędne zależności automatycznie są dołączane do tworzonego projektu.

Oprócz Spring Boot do naszego projektu użyliśmy również Bootstrapa. Bootstrap opiera się głównie na HTML, CSS i JavaScript. Zapewnia szeroką gamę wstępnie zaprojektowanych komponentów i stylów, które



Rys. 1: Architektura Spring

można łatwo dostosować do potrzeb witryny internetowej. Bootstrap jest znany ze swojej łatwości obsługi i prostoty. Zapewnia zestaw predefiniowanych klas i stylów, które można łatwo wykorzystać do tworzenia spójnych i atrakcyjnych wizualnie projektów stron internetowych.

Użycie Bootstrapa jak i Spring Boot znacznie skróciło czas pracy.

2.2.3. Spring Thymeleaf

Thymeleaf to silnik szablonów Java do przetwarzania i tworzenia HTML, XML, JavaScript, CSS i tekstu. Silnik opiera się zwykłych szablonach HTML, których podgląd można wyświetlić w przeglądarce po dwukrotnym kliknięciu, co jest bardzo pomocne przy samodzielnej pracy nad szablonami interfejsu użytkownika bez konieczności działającej serwera.

2.2.4. Spring Test

Spring Boot Test to zestaw narzędzi ułatwiający nam testowanie aplikacji napisanych z użyciem Springa. Testy w aplikacji są bardzo ważnym elementem, który zaniedbany szybko odbije się na jakości kodu i ilości błędów, które się pojawiają. Ilość pracy, którą trzeba później włożyć w naprawienie zaniedbanych testów, jest nieporównywalnie duża do czasu, który można zainwestować, tworząc je od razu. Dzięki Spring Test łatwo mogliśmy znaleźć błędy w naszym kodzie, również okazała się pomocna w utrzymaniu kodu – dają pewność, że nie zepsuliśmy czegoś przy wprowadzaniu zmian.

2.2.5. Spring Security

Spring Security to potężna i wysoce konfigurowalna platforma uwierzytelniania i kontroli dostępu. Jest to standard zabezpieczania aplikacji opartych na Springu. Spring Security to framework skupiający się na zapewnianiu zarówno uwierzytelniania, jak i autoryzacji aplikacji Java. Podobnie jak w przypadku wszystkich projektów Spring, prawdziwa siła Spring Security polega na tym, jak łatwo można go rozszerzyć, aby spełnić niestandardowe wymagania. Zapewnia między innymi kompleksową i rozszerzalną obsługę uwierzytelniania i autoryzacji, ochronę przed atakami, takimi jak utrwalanie sesji, klikanie, fałszowanie żądań między witrynami.

2.3. Oracle JDBC

JDBC to interfejs API dla języka Java, który służy do łączenia się z bazą danych. JDBC działa tak, że pozwala aplikacji uzyskać dostęp do różnych baz danych i funkcjonować na dowolnej platformie z maszyną wirtualną Java. Nie ma potrzeby pisania oddzielnych programów w celu uzyskania dostępu do różnych systemów baz danych. Użycie JDBC pozwala na napisanie jednej aplikacji, która może wysyłać polecenia SQL do różnych źródeł danych. Jest pomocna w pisaniu aplikacji w języku Java, które podłączają się do źródła danych, takiego jak baza danych.

2.4. Webjars locator

WebJars Locator to narzędzie służące do lokalizacji zasobów dostarczanych przez WebJars w aplikacjach Java. WebJars to paczki zasobów (takich jak biblioteki JavaScript, CSS, ikony, itp.), które są umieszczone w standardowych paczkach JAR i dostępne są dla projektów Java. WebJars Locator ułatwia odnajdywanie tych zasobów w kodzie aplikacji.

Podczas używania bibliotek WebJars w projekcie, WebJars Locator pozwala na dostęp do zasobów w sposób programistyczny, bez konieczności podawania dokładnej ścieżki do plików. Działa to podobnie jak mechanizm odnajdywania zasobów w zwykłych zasobach aplikacji, ale zamiast tego skonfigurowany jest do pracy z WebJars.

Dzięki WebJars Locator możesz łatwo odwoływać się do zasobów dostarczanych przez WebJars, co jest szczególnie przydatne w projektach, gdzie korzystasz z różnych bibliotek i chcesz uniknąć ręcznego zarządzania ścieżkami do plików zasobów.

2.5. Project Lombok

Project Lombok to narzędzie dla języków Java i innych, które wspiera programistów w automatyzacji generowania rutynowego kodu. Celem Lombok jest redukcja ilości kodu, który jest konieczny do napisania, ale nie wnosi wiele do samej logiki aplikacji.

Lombok osiąga to poprzez wprowadzenie adnotacji, które dodaje do kodu źródłowego, a następnie używa tych adnotacji podczas kompilacji, aby wygenerować powtarzające się fragmenty kodu. Dzięki temu, programiści mogą skupić się na logice biznesowej i zminimalizować ilość ręcznie pisanych kodu.

Przykładowe funkcje oferowane przez Lombok to:

1. @Getter / @Setter: Automatycznie generuje metody dostępne (getter i/lub setter) dla pól klasy.
2. @ToString: Automatycznie generuje metodę toString() dla klasy, opierając się na polach obiektu.
3. @EqualsAndHashCode: Automatycznie generuje metody equals() i hashCode() dla klasy, bazując na jej polach.
4. @NoArgsConstructor, @AllArgsConstructor: Generuje konstruktory bezargumentowy i/lub konstruktor, który przyjmuje wszystkie pola klasy.
5. @Builder: Ułatwia tworzenie wzorca projektowego Builder, co ułatwia tworzenie obiektów z wieloma opcjonalnymi parametrami.

3. Działanie aplikacji

Podczas pracy nad drugą częścią projektu stworzyliśmy aplikację webową. Aplikacja pozwala na logowanie się na jeden z dwóch profili, jako użytkownik-klient oraz jako administrator. Każdemu daje możliwość innych funkcji i ruchów na stronie. Aplikacja uwzględnia błędy wpisywanych danych, takich jak login, hasło czy też błędy wynikające z prób naruszenia więzów integralności bazy podczas usuwania, czy też edycji danych.

Projekt miał spełniać wymagania trzech płaszczyzn.

3.1. Pierwsza płaszczyzna Aplikacji

3.1.1. Intuicyjność

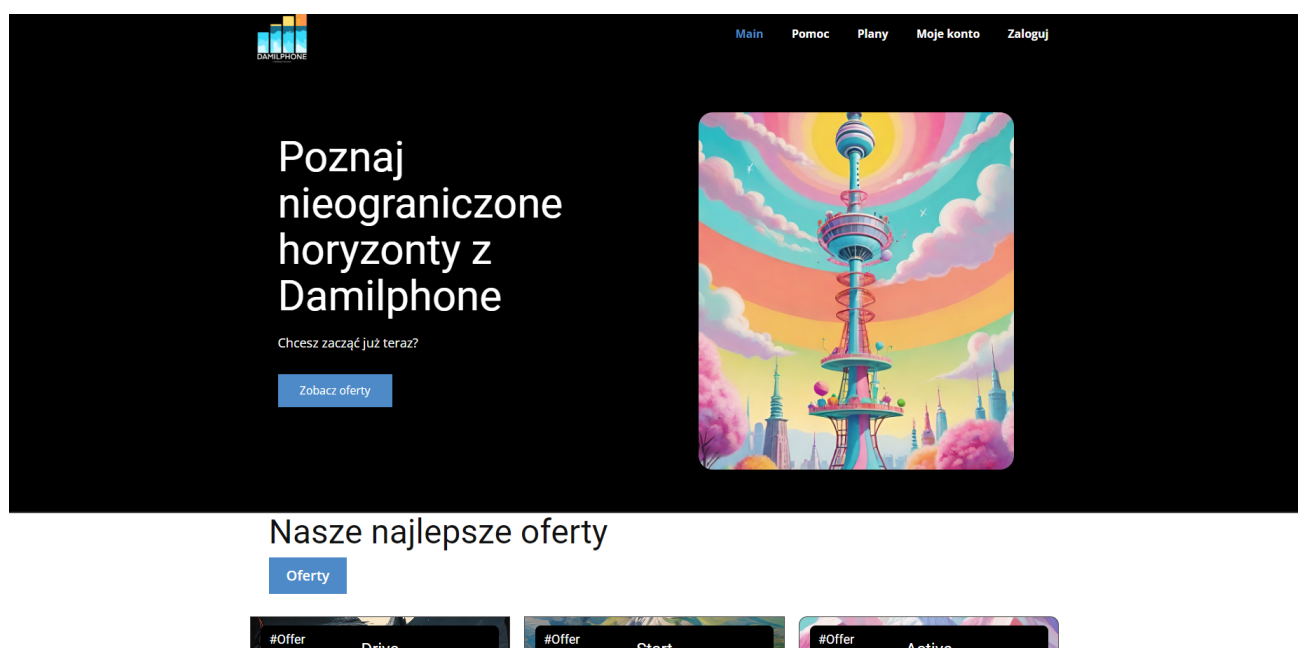
Nasza aplikacja spełnia założenia dotyczące intuicyjności, która wpływa na użyteczność i doświadczenie użytkownika, dzięki kilku aspektom:

1. jasna nawigacja: główne menu nawigacyjne (rys. 2) jest logicznie zorganizowane, umożliwiając użytkownikowi szybkie odnalezienie potrzebnych treści. Nawigacja ta jest spójna na wszystkich stronach.



Rys. 2: Menu nawigacyjne

2. prostota strony głównej (rys. 11) i wyróżnienie najistotniejszych informacji.



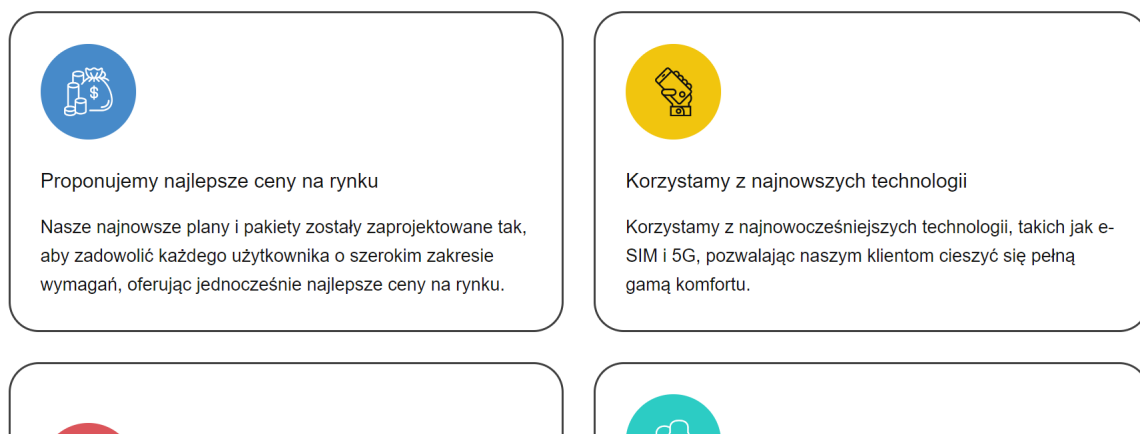
Rys. 3: Strona główna

3.1.2. Przyjrzyście

Przejrzystość aplikacji to kluczowy element, który wpływa na zadowolenie użytkowników. Interakcja z aplikacją jest klarowna, zrozumiała i przyjazna dzięki:

1. czystości projektu graficznego i klarowności informacji uzyskanej przez dostosowanie sposobu prezentacji, jak np. sekcja "Czemu my?" (rys. 4), która w spójny i przyjazny sposób zachęca do skorzystania z usług naszego operatora

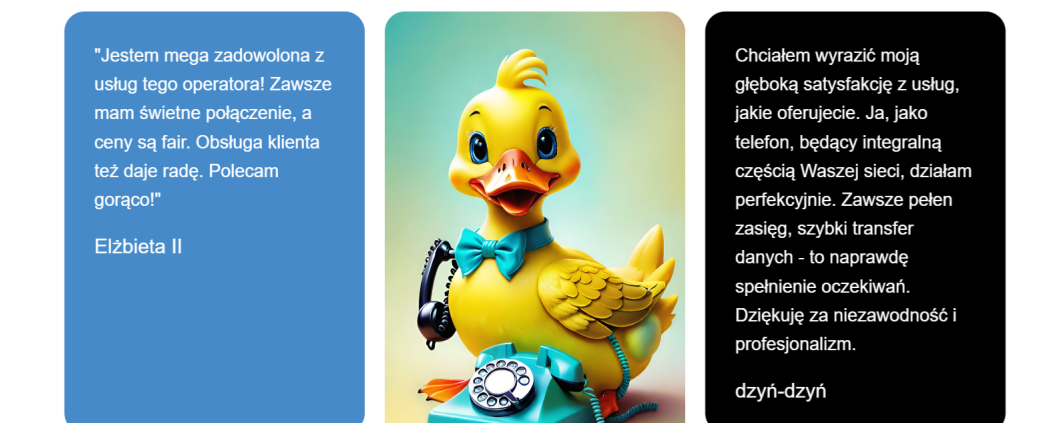
Czemu my?



Rys. 4: Sekcja "Czemu my?"

2. ciekawemu interfejsowi użytkownika poprzez zastosowanie odpowiedniej i ciekawej kolorystyki, jak np. w sekcji "Opinie naszych klientów" (rys. 5).

Opinie naszych klientów



Rys. 5: Sekcja "Opinie naszych klientów"

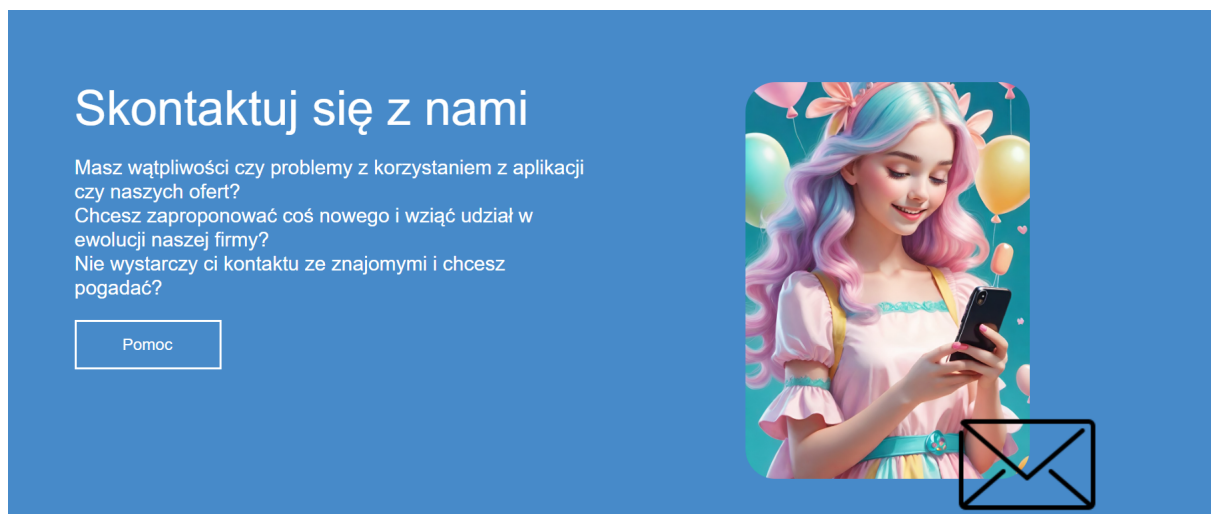
3.1.3. Pomysł

Nasz pomysł na uatrakcyjnienie strony naszego operatora to między innymi:

1. istnienie dokumentacji: dostarczenie łatwo dostępnej dokumentacji i FAQ (-_-) (rys. 6) czy pomocy (rys. 7) może pomóc użytkownikom w zrozumieniu funkcji i rozwiązaniu ewentualnych problemów.



Rys. 6: Regulamin naszego operatora



Rys. 7: Kontakt i pomoc

✉ **NAPISZ DO NAS**

Nasz spójny zestaw zawsze jest gotowy ci pomóc

hi@ourcompany.com

📍 **BIURO GŁÓWNE**

Przyjdź i powiedz "cześć"

Krasnaya sq., 1, Moscow, Russia, 109012

☎ **TELEFON**

Poniedziałek-sobota 8:00-22:00

+48 555 555 555

Skontaktuj się z nami

Podrobnie opisując swój problem.

Imię

Email

Jak możemy ci pomóc?

WYŚLIJ

Rys. 8: Kontakt - formularz

Formularz na rysunku 8. faktycznie przesyła na naszego maila informacje wpisane przez użytkownika.

2. Zastosowanie interaktywnych elementów na stronie.

Jak sprawdzić swoją aktualną ofertę oraz plan taryfowy?

+

Gdzie znaleźć wszystkie dostępne oferty?

+

Możesz sprawdzić aktualne plany na naszej stronie w zakładce "Plany"

POKAŻ

Czy mogę zmienić swoje dane osobowe?

+

Czy mogę założyć konto, nie będąc klientem Damilphone?

+

Jak dbacie o prywatność danych użytkowników?

✓

Jeśli Twoje pytanie nie znalazło odpowiedzi powyżej, skontaktuj się z naszym działem Obsługi Klienta infolinię lub adres mailowy. Jesteśmy tutaj, aby pomóc! Dziękujemy, że jesteś z Damilphone.

+48 555 555 555

Szybkie odpowiedzi na często zadawane pytania

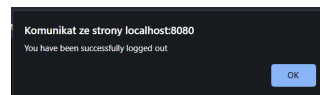
Rys. 9: FAQ

3. Komunikacja w przypadku, gdy użytkownik wprowadził dane, które mogą prowadzić do błędu wewnętrznego czy naruszają reguły biznesowe bazy danych.



Rys. 10: ErrorMessage: "Wprowadź wymagany format"

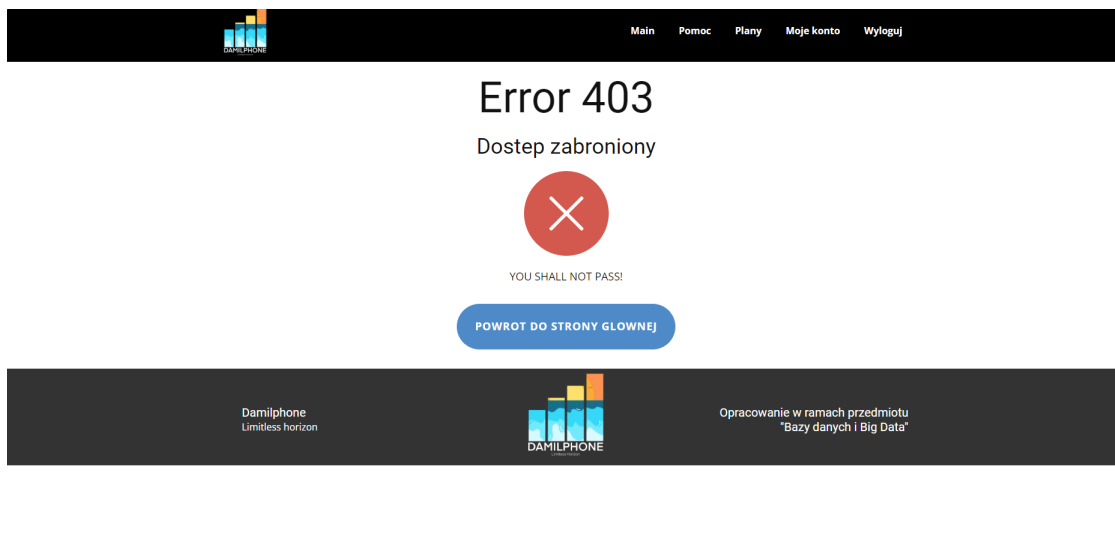
4. dymek informacyjny o poprawnym wylogowaniu



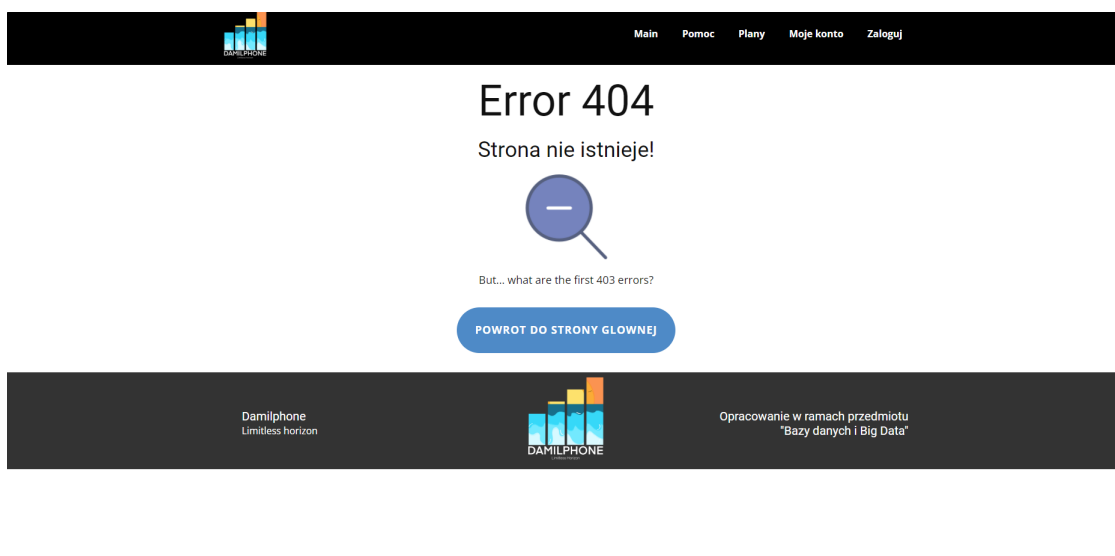
Rys. 11: Alert o poprawnym wylogowaniu

3.1.4. Jakość rozwiązania i obsługa błędów

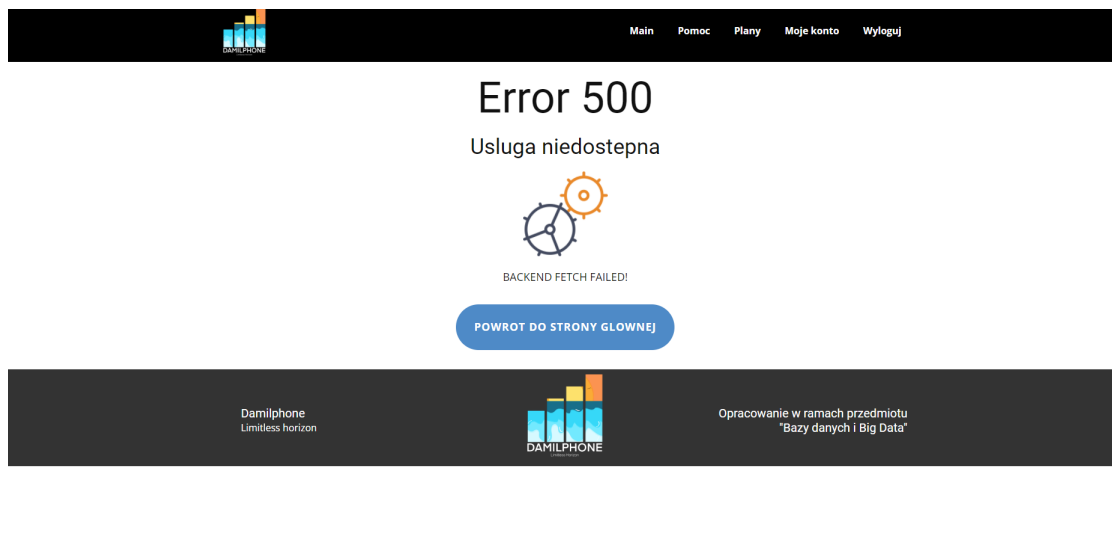
Aby zapewnić wysoką jakość rozwiązania nasza aplikacja gwarantuje obsługę błędów, np:



Rys. 12: Error 403 - Dostęp zabroniony



Rys. 13: Error 404 - Strona nie istnieje



Rys. 14: Error 500 - Usługa niedostępna

3.2. Druga płaszczyzna Aplikacji

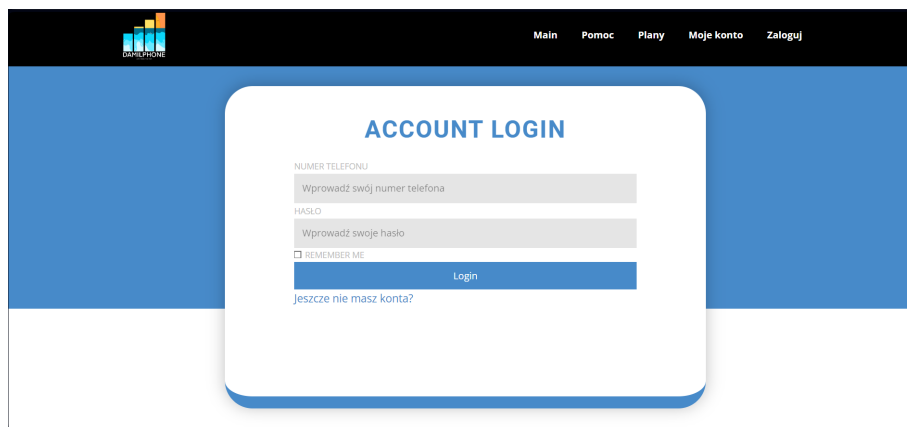
3.2.1. Perspektywa użytkownika

Użytkownik na stronie naszego operatora ma dostęp do strony głównej i jego podstron. Dodatkowo jako klient może się zarejestrować, aby uzyskać dostęp do swojego konta z danymi lub po prostu się zalogować, jeżeli posiada już konto. Po przejściu na konto może przeglądać swoje dane podstawowe i je edytować oraz przeglądać swoje umowy.

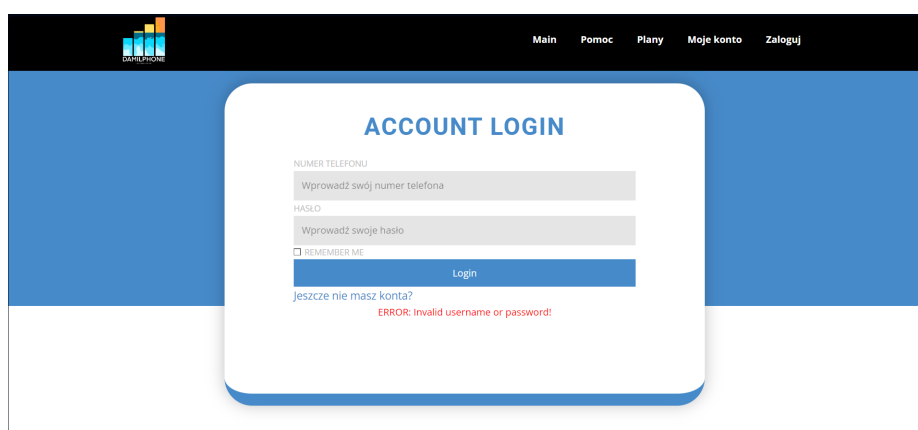
1. Ważną częścią perspektywy wąskiej jest możliwość rejestracji nowego użytkownika (rys. 15). Skrypt sprawdza, czy użytkownik wprowadził numer telefonu poprawnego formatu oraz, czy istnieje numer telefonu w bazie, jako telefon naszego klienta. W przypadku jeżeli jakieś z tych warunków nie jest spełniony użytkownik dostaje odpowiedni komunikat. W razie sukcesu nowy user jest automatycznie dodany do tabeli "Uzytkownicy", która jest podłączona do bazy.

Rys. 15: Rejestracja

2. Login (rys. 16). Wraz ze zrunowaniem aplikacji, w klasie SpringSecurity tworzą się uprawnienia dostępu. rola 'ADMIN' jest stała, natomiast rola 'USER' nadaje się wszystkim użytkownikom, znajdującym się w tabeli "Uzytkownicy", każdemu z nich oprócz roli 'USER' jest przyswojony unikatowy ID, który z kolei jest równy kluczowi głównemu "id_uzytkownika". Podczas zalogowania, jest sprawdzone, czy wprowadzone dane istnieją w konfiguracji bezpieczeństwa. W przypadku gdy użytkownik nie ma uprawnień do zalogowania się, on dostaje odpowiedni komunikat (rys. 17).

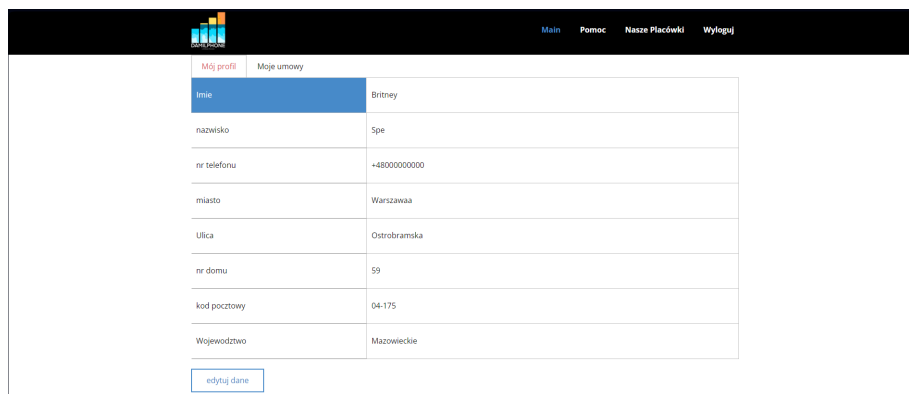


Rys. 16: Strona logowania



Rys. 17: Potwierdzenie bycia klientem operatora

3. Każdy użytkownik zalogowany z rolą 'USER' może wejść na własne konto i przejść do przeglądu i modyfikacji danych. W zakładce "Mój profil" użytkownik widzi swoje dane osobowe (Imię, Nazwisko, Adres itd) (rys. 18). Po wciśnięciu przycisku "edytuj dane" użytkownika przenosi na stronę modyfikacji (rys. ??). Po wpisaniu nowych danych przez użytkownika i dość skomplikowanych operacjach backend'owych, tabele "Klienci", "Adresy" i "Poczty" się aktualizują. W zakładce "Moje umowy" klient widzi poppisane przez niego umowy i informację ich dotyczącą (Dane są pobierane z różnych tabel: "Umowy", "Oferty", "Placówki") (rys. 23).



Imię	Britney
nazwisko	Spe
nr telefonu	+48000000000
miasto	Warszawa
Ulica	Ostrobramska
nr domu	59
kod pocztowy	04-175
Województwo	Mazowieckie

[edytuj dane](#)

Rys. 18: Przegląd swojego profilu przez użytkownika



Nazwa oferty	Data podpisania	Okres trwania (miesiące)	Adres mailowy placówki
Active	1234-12-11	6 month	polna@osk.pl

Damilphone
Limitless horizon

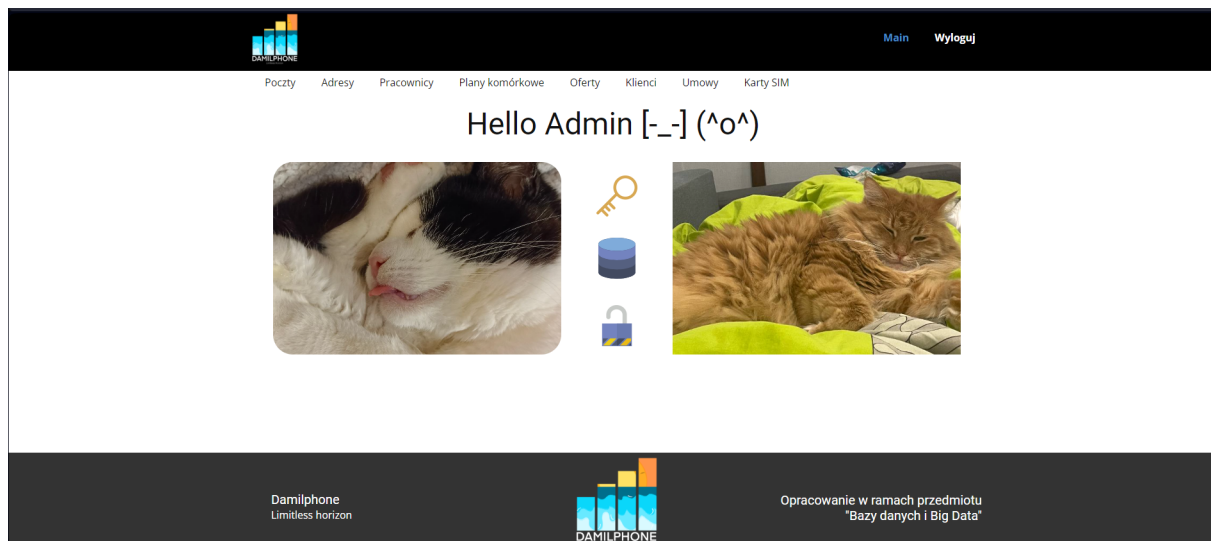
Opracowanie w ramach przedmiotu
"Bazy danych i Big Data"

Rys. 19: Przegląd podpisanych umów

3.2.2. Perspektywa administratora

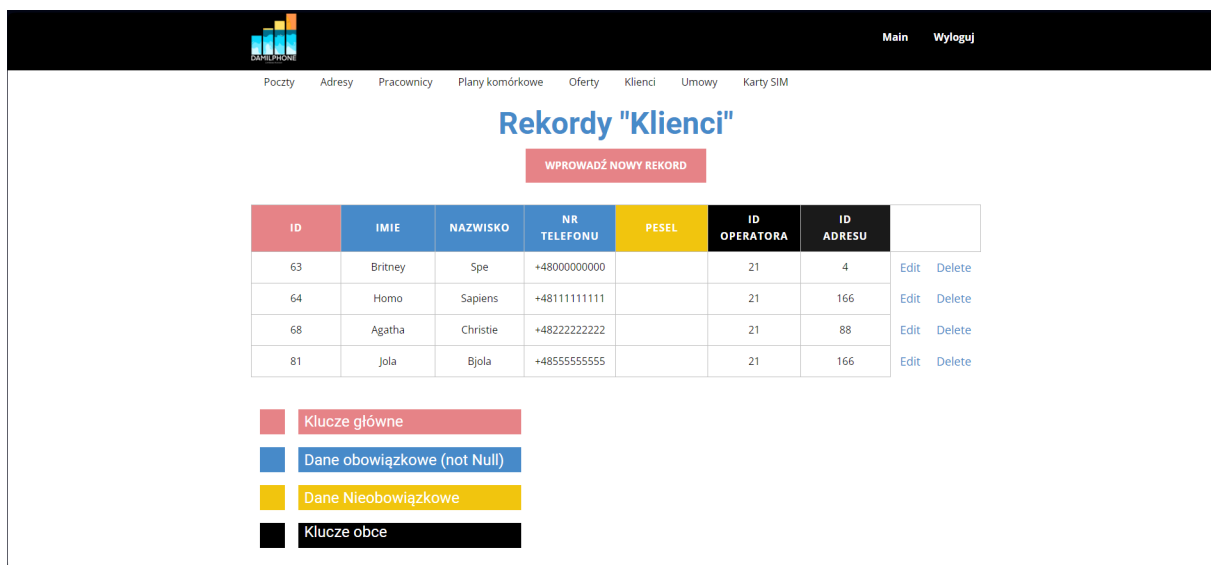
Administrator podobnie jak klient na stronie głównej musi się zalogować, aby uzyskać dostęp do stron administracyjnych. Po udanym zalogowaniu jest w stanie przeglądać, wprowadzać nowe rekordy, edytować istniejące rekordy i je usuwać wybranym tabelom widniejącym na górnym pasku.

1. Strona administratora



Rys. 20: Widok administratora - strona główna

2. Przegląd tabel wraz z podglądem tabel pomocniczych




Rys. 21: Widok administratora - Przegląd danych z tabeli "Klienci"

Tabela pomocnicza "Operatorzy"			
ID	NAZWA	DATA ZAŁOŻENIA	ID ADRESU
21	Damilphone	2024-01-21 19:21:49	88


Tabela pomocnicza "Adresy"				
ID ADRESU	MIASTO	ULICA	NR_DOMU	ID_POCZTY
166	Warszwa	Waryńskiego	13	85
167	Warszawa	Pomocznica	12	86
88	Warszawa	Polna	12	84
4	Warszawaa	Ostrobramska	59	6

Rys. 22: Tabele pomocnicze - "Operatorzy", "Adresy"


3. Dodawanie nowych rekordów. Po wpisaniu danych i wcisnięciu przycisku "OK", jest wywołana funkcja JavaScript, sprawdzająca zgodność wprowadzonych danych. W przypadku ich niezgodności, albo wpisaniu nie istniejącego klucza obcego widzimy dymkę informacyjną (rys. ??)



Hint: Użyj Wymaganych formatów (Są podane przy nazwie atrybutu)



Hint: Skorzystaj z kolorystycznej podpowiedzi pod tabelą



Hint: Dla czarnych pól użyj wartości z tabeli pomocniczych na dole

Imie (String[20])	<input type="text" value="Britney"/>
Nazwisko (String[30])	<input type="text" value="Spe"/>
Pesel (String[11])	<input type="text"/>
Numer Telefonu (String[12](format: +48xxxxxxxxxx))	<input type="text" value="+48000000000"/>
ID Operatora (int)	<input type="text" value="21"/>
ID Adresu (int)	<input type="text" value="4"/>


ZACHOWAJ


Klucze główne


Dane obowiązkowe (not Null)

Rys. 23: Widok administratora - Dodawanie nowych rekordów

4. Edycja rekordów (rys. 24) Również jest sprawdzona na zgodność - tylko potem update'owana.

 Hint: Użyj Wymaganych formatów (Są podane przy nazwie atrybutu)

 Hint: Skorzystaj z kolorystycznej podpowiedzi pod tabelą

 Hint: Dla czarnych pól użyj wartości z tabeli pomocniczych na dole

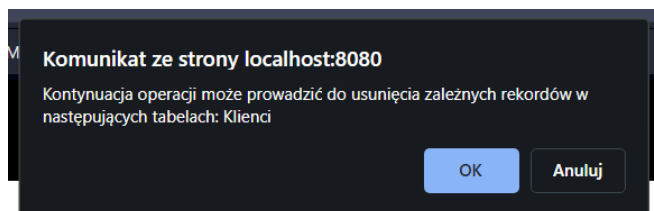
Imie (String[20])	<input type="text" value="Britney"/>
Nazwisko (String[30])	<input type="text" value="Spe"/>
Pesel (String[11])	<input type="text"/>
Numer Telefonu (String[12](format: +48xxxxxxxxxx))	<input type="text" value="+48000000000"/>
ID Operatora (int)	<input type="text" value="21"/>
ID Adresu (int)	<input type="text" value="4"/>

ZACHOWAJ

Klucze główne
Dane obowiązkowe (not Null)

Rys. 24: Widok administratora - Edycja nowych rekordów

5. Usunięcie rekordów. W przypadku gdy pole rekordu występuje w niektórych rekordach zależnych jako klucz obcy, user dostaje komunikat z pytaniem czy na pewno chce kontynuować operację (rys. 25). W przypadku gdy użytkownik zaznaczy opcję "OK" wywołuje się operacja usunięcia kaskadowego. Możliwość tej operacji została dodana w narzędziu SQL Developer. Zmieniliśmy kod kluczy obcych poleceniem ON DELETE CASCADE, to umożliwiło usunięcie danego rekordu, wraz z rekordami zależnymi (Działa jako drzewko: według jednej gałęzi usuwa wszystko dopóki nie wystąpi rekord od którego nic nie zależy).



Rys. 25: Usunięcie kaskadowe -