Broker ofert miejsc noclegowych

Etapy tworzenia projektu:

1. Analiza wymagań.

Podczas analizy wymagań zdefiniowaliśmy cztery moduły, które należało zaimplementować. Główny **Premise Broker** i trzy pomocnicze **Bidder System, Bank System, Debt Collector System**.

- **Premise Broker:**

Główny moduł odpowiedzialny za zbieranie oferty noclegowych od offerentów (bidder), synchronizowanie ofert, rezerwacji z innymi webserwisami takimi jak **Bidder System** oraz udostępnianie profilu użytkownika.

- **Bidder System:**

Moduł pomocniczy służący do zobrazowania możliwości synchronizacji danych z modułem głównym.

Po analizie wymagań zdecydowaliśmy, że moduły **Premise Broker** oraz **Bidder System** zostaną zaimplementowane zgodnie z architekturą REST, a moduły **Bank System, Debt Collector System** z SOAP.

3. Zdefiniowanie usług:

- **Premise Broker**

Ścieżka	GET	POST	PATCH	DELETE
/api/users	-	+	-	-
/api/bidders	-	+	-	-
/api/users/userId/bidder	+	-	-	-
/api/premises	+	+	-	-
/api/premises/premiseId	+	-	+	+
/api/offers/offerId/premises	+	-	-	-
/api/offers	+	+	-	-
/api/offers/offerId	+	-	+	+
/api/premiseReservations	+	+	-	-
/api/premiseReservations/resId	+	-	+	-
/api/premiseReservations/resId /premise	+	-	-	-
/api/premiseReservations/search/bidder ?bidderId=bidderId	+	-	-	-
/api/users/userId/premiseReservations	+	-	-	-
/api/favouriteOffers	+	+	-	-
/api/users/userId/favouriteOffers? projection=favouriteOffer	+	-	-	-
/wire/notification	-	+	-	-

Nazwy większości usług tłumaczą za co te usługi są odpowiedzialne.

Ostatnia usługa **/wire/notification** odbiera informację z banku, że otrzymaliśmy przelew .

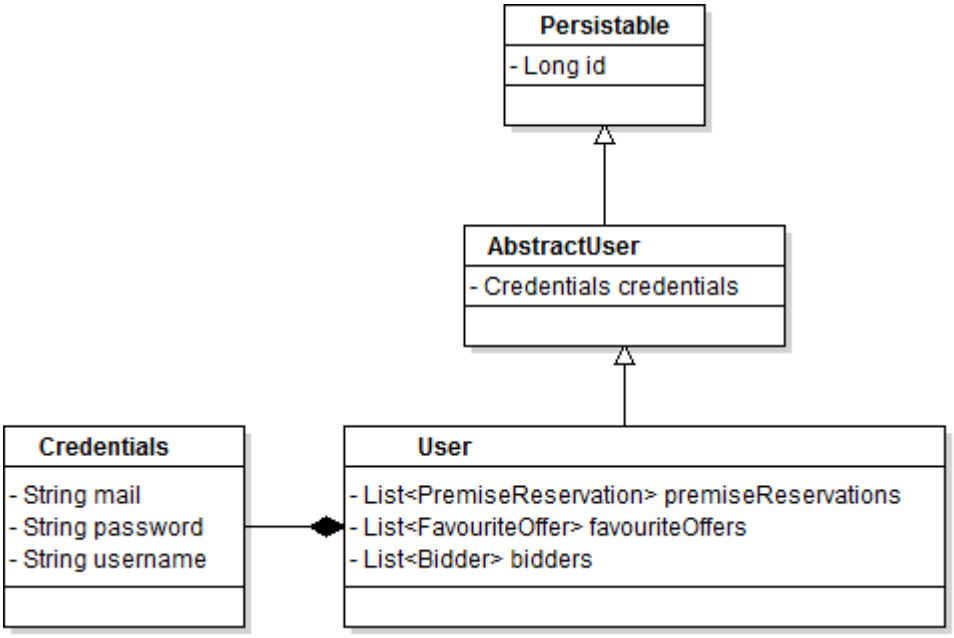
- **Premise Broker**

Ścieżka	GET	POST	PATCH	DELETE
/api/users	-	+	-	-
/api/premises	+	+	-	-
/api/premises/premiseId	+	-	+	+
/api/offers/offerId/premises	+	-	-	-
/api/offers	+	+	-	-
/api/offers/offerId	+	-	+	+
/api/premiseReservations	+	+	-	-
/api/premiseReservations/resId	+	-	+	-
/api/premiseReservations/resId /premise	+	-	-	-
/api/users/userId/premiseReservations	+	-	-	-
/invoice	-	+	-	-
/foreignReservation	-	+	+	-

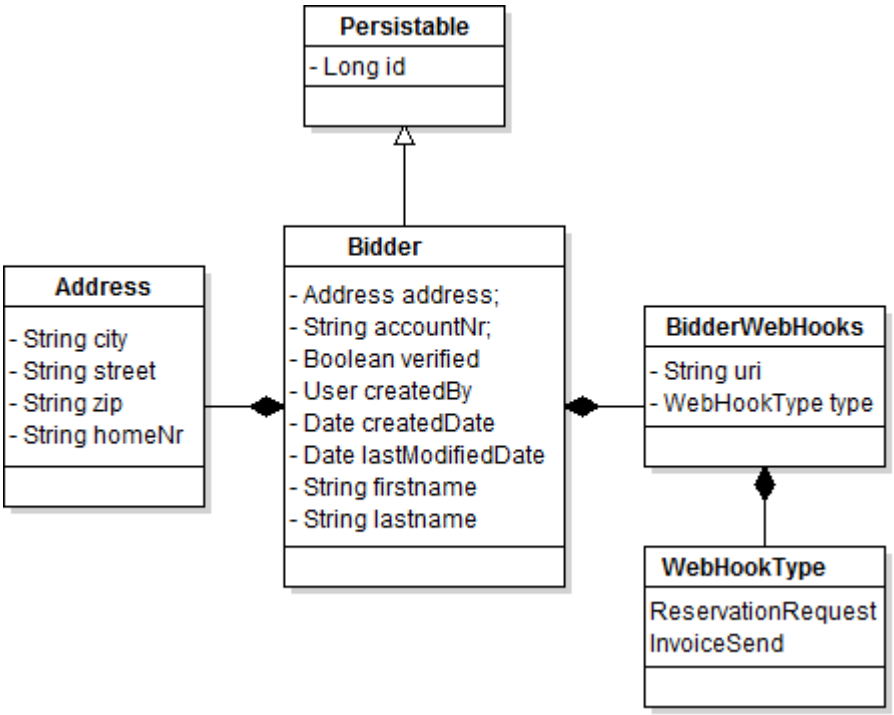
Usługa **/foreignReservation** odpowiedzialna jest za obsługę zapytań od **Premise Brokera** dotyczących rezerwacji miejsc.

4. Projekt warstwy model

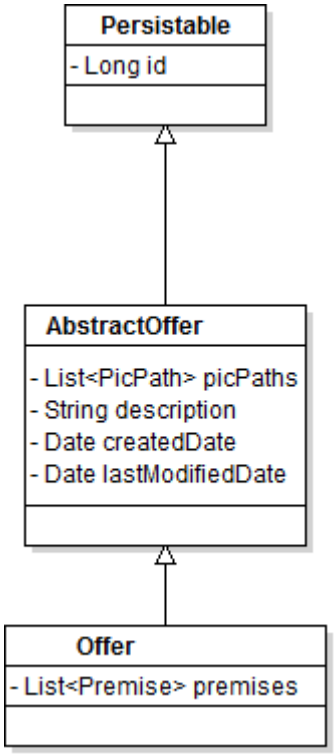
- **User**



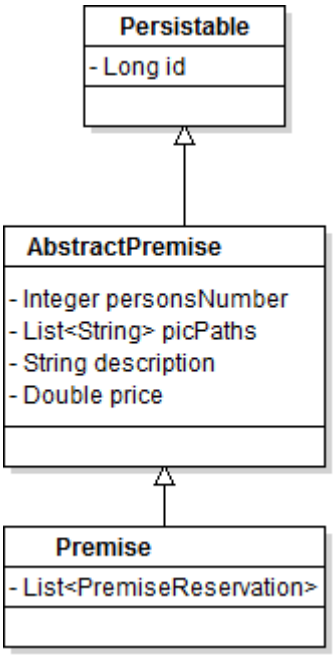
- **Bidder**



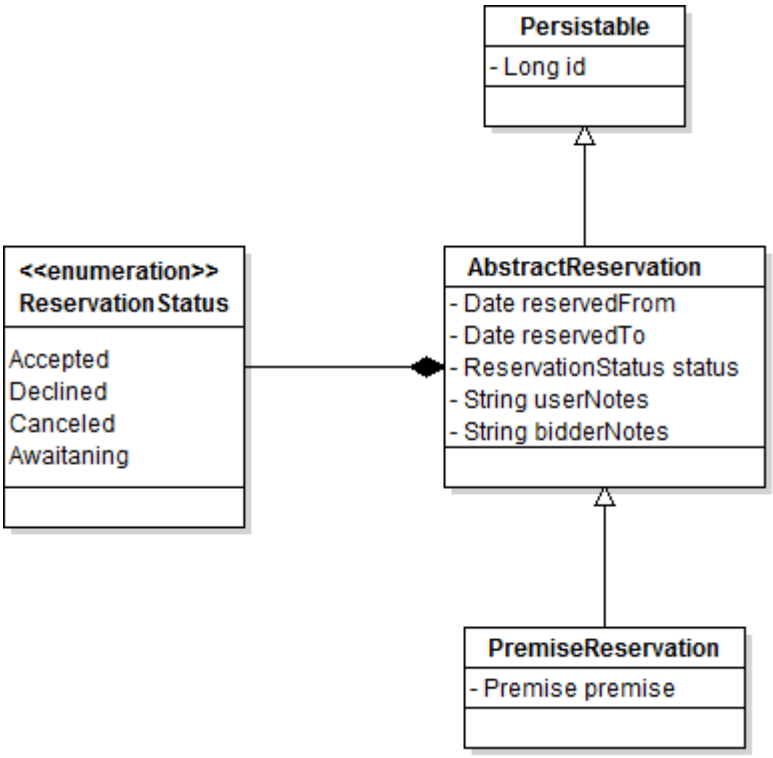
- **Offer**



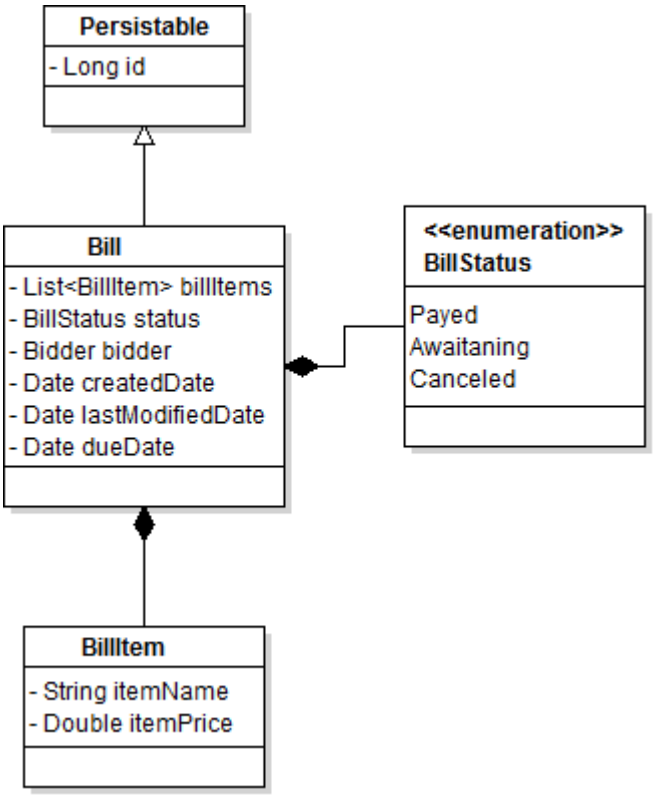
- **Premise**



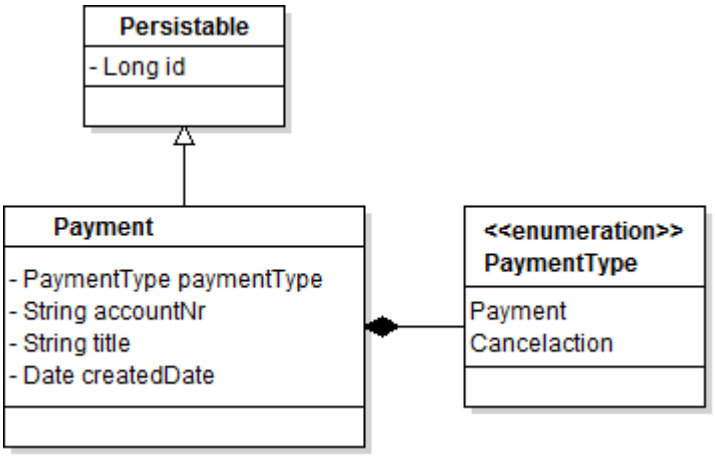
- **PremiseReservation**



- **Bill**



- **Payment**



5. Szczegóły implementacji

- **Konfiguracja**

Parametry(loginy i hasła do systemów zewnętrznych, adres bazy danych, cron expressions itd.) do konfiguracji systemów przechowywane są w plikach application.properties.

- **Bezpieczeństwo**

W modułach **Premise Broker** oraz **Bidder System** za bezpieczeństwo odpowiada biblioteka spring-security. Autentykacja użytkownika jest zaimplementowana zgodnie z Basic Authentication. Z powodu tego, iż większość zapytań wymaga autoryzacji, a architektura REST nie pozwala na przechowywania jakiegokolwiek stanu(sesji, ciasteczek) po stronie serwera, wprowadziliśmy wymaganie, aby wszystkie zapytania zawierały nagłówek Authorization. W każdym zapytaniu wysyłamy login i hasło użytkownika, dlatego też wszystkie usługi są zabezpieczone certyfikatem SSL.

Pierwsza autoryzacja użytkownika w większości przypadków wykonywana jest zaraz na początku przetwarzania zapytania i polega na sprawdzeniu czy dany użytkownik jest tym za kogo się podaję np. przy wywołaniu usługi **/api/users/userId/premiseReservations** sprawdzamy czy "zalogowany" użytkownik ma nr id taki sam jak userId z zapytania. Dodatkowa autoryzacja wykonywana jest przed zapisem do bazy jeśli jest to konieczne.

- **Komunikacja z zewnętrznymi systemami**

Do komunikacji z zewnętrznymi typu **Bidder System** użyliśmy RestTemplate. Wywoływane adresy URI przechowywane są w tabelce **BidderWebHooks.** Komunikacja z **Bank System** oraz **Debt Collector System** zaimplementowana jest przy użyciu Apache Axis.

- **Automatyczne wykonywanie zadań**

Zadania : wysyłanie miesięcznych faktur, wysyłanie informacji o rezerwacjach, zgłoszenie do windykatora są wykonywane automatycznie przy użyciu biblioteki Quartz. Czas i częstotliwość ww. zadań określana jest przez cron expressions.

- **Szczegóły implementacji usług**

Większość usług została zaimplementowana przy użyciu biblioteki spring-data-rest, która obsługuję m. in. serializację i deserializację danych, odczyt i zapis do bazy oraz udostępnia obsługę zdarzeń przed i po zapisie do bazy (HandleBeforeCreate, HandleBeforeSave, HandleBeforeDelete, HandleAfterCreate, HandleAfterSave, HandleAfterDelete.)

Premise Broker

- **Offer** (HandleBeforeCreate, HandleBeforeSave, HandleBeforeDelete)
    1. Autoryzacja użytkownika (czy potwierdził swoje dane przelewem)
- **Bidder** (`HandleAfterCreate`)
    1. Stworzenie rachunku i wysłanie faktury na maila bądź przez usługę finansową **Bidder Systemu**.
- **PremiseReservation** (HandleAfterCreate)
    1. Wysłanie prośby o potwierdzenie rezerwacji do **Bidder System**
    2. Uaktualnienie rezerwacji o otrzymany status.
- **PremiseReservation** (HandleBeforeSave)
    1. Autoryzacja
    2. Wysłanie prośby o potwierdzenie rezerwacji do **Bidder System**
    3. Uaktualnienie rezerwacji o otrzymany status.
- **/wire/notification**
    1. Wysłanie zapytania do **Bank System** o potwierdzenie przelewu, na podstawie otrzymanego id.
    2. Utworzenie rekordu w tabelce **Payment** i zaktualizowanie rekordu **Bill**
    3. Jeśli był to przelew potwierdzający rejestrację **Bidder** aktualizujemy rekord **Bidder**

Bidder System

- **Offer** (HandleAfterCreate, HandleAfterSave, HandleAfterDelete)
    1. Wysłanie informacji do **Bidder System** o stworzeniu, zmianie, usunięciu **Offer**.
- **Premise** (HandleAfterCreate, HandleAfterSave, HandleAfterDelete)
    1. Wysłanie informacji do **Bidder System** o stworzeniu, zmianie, usunięciu **Premise**.
- **/invoice**
    1. Wysłanie przelewu przy użyciu usługi **Bank System**
- **/foreignReservation**
    1. Akceptacja bądź odrzucenie rezerwacji
    2. Zapis do bazy
    3. Zwrócenie statusu rezerwacji wraz z uwagami.