Wybrany serwer aplikacyjny i bazodanowy

W ramach migracji wybrany został Quarkus i MariaDB.

Quarkus – nowoczesny, zorientowany na wdrożenie w chmurze, Kubernetes-native szkielet aplikacyjny Java. Zaimplementowany pod użycie z natywnymi obrazami GraalVM/HotSpot.

MariaDB – system zarządzania relacyjną bazą danych, open-source. Powstał z forka MySQL. Zorientowany jest na wydajność i skalowalność.

Opis zmian w konfiguracji serwera aplikacyjnego, o ile takie były potrzebne

Konfiguracja jednostek składowania w dedykowanych profilach application.properties zamiast w persistence.xml.

Opis procesu generowania struktur w bazie danych
Migracja truncate w test-init.sql na inne polecenia
Dostosowanie składni w init.sql i docker-init.sql
Zamiana sekwencji na AUTO_INCREMENT
Zmiana nazw kolumn – niektóre posiadały zarezerwowane nazwy „year_month”
migracja funkcji bazodanowej DATE_PART na MONTH;

Opis pozostałych czynności administracyjnych w bazie danych (nakładanie ograniczeń, uprawnień itp.)

Migracja mechanizmu czyszczenia testów - zamiast batch sql, po kolei;

Opis zmian w konfiguracji projektów

Dodanie konfiguracji uruchomieniowej dla quarkus
Usunięnie konfiguracji uruchomieniowej Payara

Opis zmian kodu Java, o ile było to wymagane

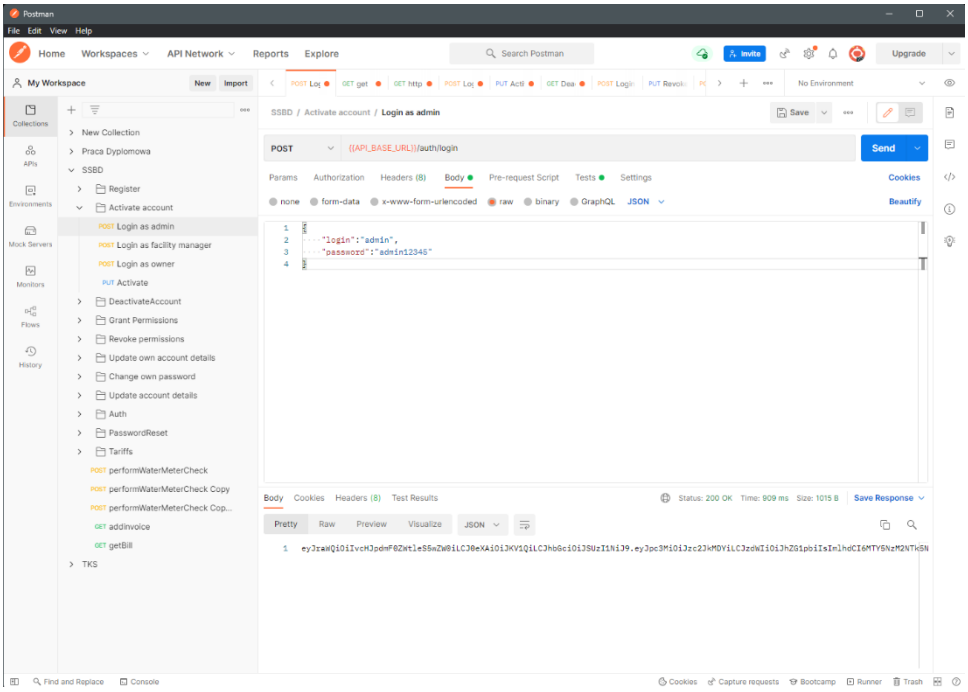
zmiana konfiguracji dla Testcontainers – dostosowano do MariaDB @QuarkusTestResource
dodanie mechanizmu obsługi testów @QuarkusTest
poprawa formatu daty w testach;
Zmiana regexp dla constraintów – inne logi z constraintów bazodanowych;
fix konfiguracji dla testowej mariaDB;
usunięcie z importów z pakietu ejb, zamiana komponentów ejb na CDI
Usunięto @TransactionAttributes, zamieniono na @Transactional na controllerach restowych, tak aby propagowane były one do kolejnych komponentów.
Zmiana mechanizmu uwierzytelniania oraz generowania tokenu jwt. Ponieważ quarkus nie wykorzystywał HttpAuthenticathionMechanism z pakietu jakarta.security. Zamiast tego skorzystaliśmy z gotowej implementacji smallrye-jwt dostosowanej do quarkusa, wymagane było dodanie odpowiedniego dependency oraz konfiguracja (w tym wygenerowanie pary kluczy do podpisywania i weryfikacji integralności tokenu)

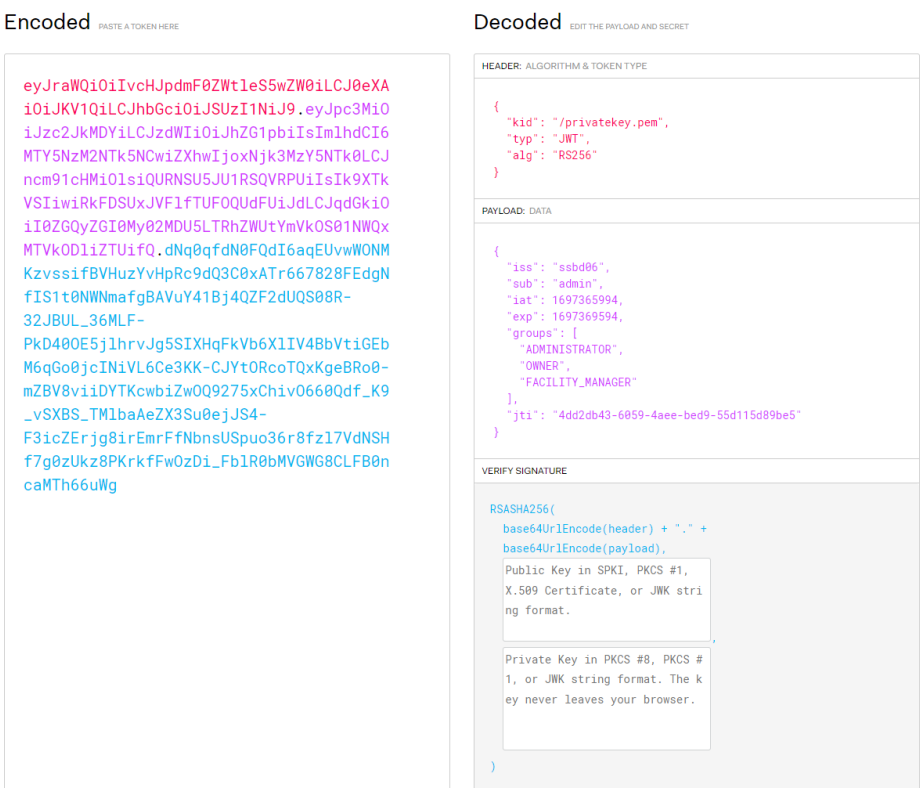
Zmiana email providera na quarkus-mailer

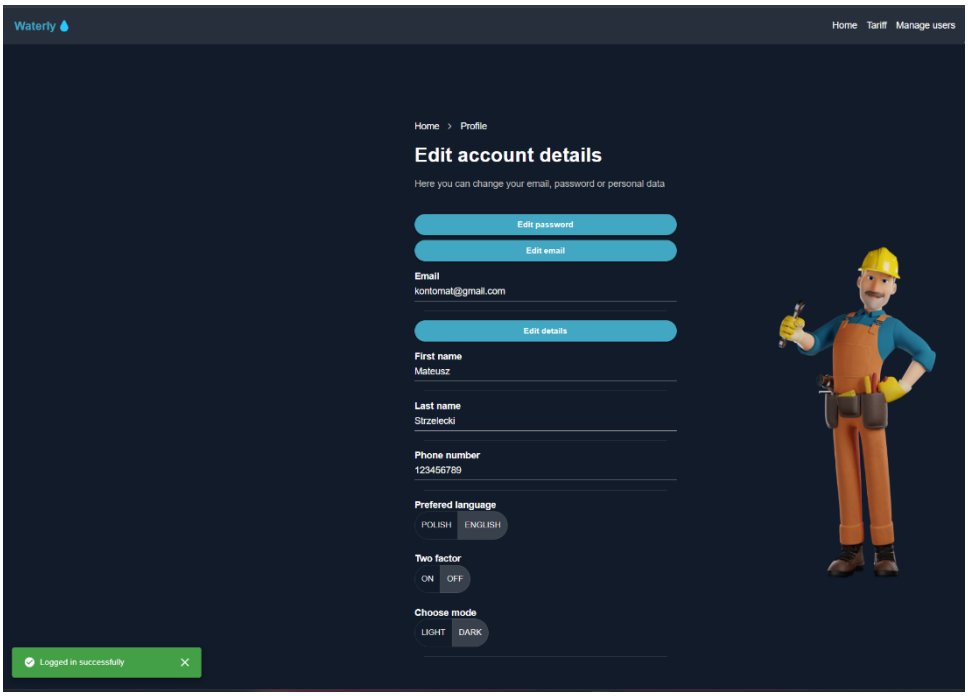
Zmiana etag filtra, nie dało się injecotwac httprequesta do interceptora, teraz wyciagamy headery z singletona Arc

Lekko zmieniona struktura jwt, dostosowanie na front endzie.

Prezentacja (zrzuty ekranu) prawidłowego procesu uwierzytelniania we wdrożonej aplikacji







Instrukcja uruchomienia:

W projekcie istnieje konfiguracja uruchomieniowa quarkus. Wystarczy zaimportować ją do IntelliJ. Ewentualnie należy zbudować projekt za pomocą mvn clean install, a następnie za pomocą quarkus-cli:
quarkus dev
Ewentualnie można skorzystać z załączonego jara:
java -D"quarkus.profile"=dev -jar .\target\ssbd06-runner.jar
lub przygotowanego skryptu mvn:
./mvnw quarkus:dev -DskipTests

Przed uruchomieniem aplikacji, należy postawić serwer bazodanowy. Konfiguracja znajduje się w projektowym docker-compose, w katalogu local.

```
Running 219/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$CreateInvoice#CreateInvoice
Running 219/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$CreateInvoice#shouldFailCreateInvoiceWhenThereAreNoWaterMeterMeterMeterCheck()
Running 220/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$CreateInvoice#shouldCreateInvoice()
Running 221/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$CreateInvoice#shouldForbidInvoiceCreationWhenDataIsInvalid(String, BigDecimal)
Running 221/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$CreateInvoice#date: null, totalCost: 100.00
Running 222/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$CreateInvoice#date: 2043-10-10, totalCost: null
Running 223/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$CreateInvoice#shouldFailCreateInvoiceWhenThereIsNoTariff()
Running 224/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$CreateInvoice#shouldFailCreateInvoiceWhenDateIsInvalid()
Running 225/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$CreateInvoice#shouldFailCreateInvoiceWhenInvoiceNumberIsInvalid()
Running 226/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$UpdateInvoice#UpdateInvoice
Running 226/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$UpdateInvoice#shouldForbidInvoiceUpdateWithNoAuthorization()
Running 227/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$UpdateInvoice#shouldUpdateInvoice()
Running 228/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$UpdateInvoice#shouldForbidInvoiceUpdateWhenDateIsColliding()
Running 229/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$UpdateInvoice#shouldForbidInvoiceUpdateWhenDataIsInvalid()
Running 230/126. Running: pl.lodz.p.it.ssbd2023.ssbd06.integration.mol.InvoiceControllerTest$UpdateInvoice#shouldForbidInvoiceUpdateForNotFacilityManager()
Press [e] to edit command line args (currently ''), [r] to re-run, [h] for more options>
Press [e] to edit command line args (currently ''), [r] to re-run, [o] Toggle test output, [h] for more options>
All 229 tests are passing (1 skipped), 229 tests were run in 323008ms. Tests completed at 13:10:22.
```