**GROUP SCHEDULE**



Dokumentacja techniczna

AUTORZY:

Ogórek Zbigniew

Sądel Bartosz

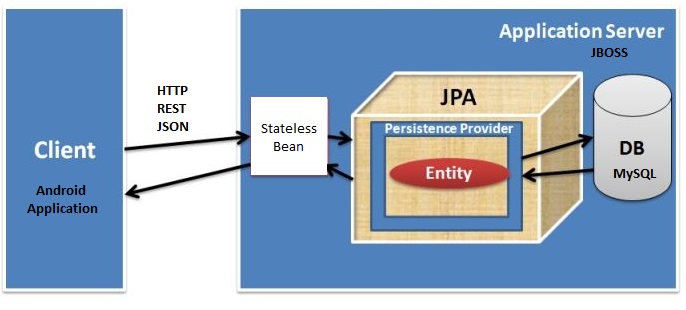
**AGH**

Wydział Informatyki, Elektroniki i Telekomunikacji

1. **Wstęp**

Projekt można podzielić na dwie odrębne części, z których jedna zależy od drugiej. Pierwsza z nich to część serwerowa naszej aplikacji. Druga odpowiada za uruchomienie aplikacji na konkretnej platformie, w tym przypadku na urządzeniu mobilnym z oprogramowaniem Android.

Poniżej przedstawiony jest podstawowy schemat:



W kolejnych sekcjach dokumentu zostanie szerzej opisana struktura poszczególnych modułów.

1. **Wykorzystane narzędzia i technologie**

Narzędziami które wykorzystaliśmy do stworzenia naszego projektu były:

**Android studio** – Środowisko programistyczne oferujące wszystkie podstawowe pomoce, takie jak kolorowanie oraz podpowiadanie składni. Zostało ono przez wybrane głównie z powodu dużej liczby dostępnych wtyczek, w tym wspierających pliki budujące Gradle, pliki konfiguracyjne czy skrypty Shella.

**Intellij IDEA** – Środowisko programistyczne wspomagające prace przy tworzeniu aplikacji typu Java Enterprise. Posiada podstawowe funkcja takie jak kolorowanie i podpowiadanie składni oraz bardziej zaawansowane, jak integracja z aplikację EJB, deploy i debugowanie na lokalny i zdalnym serwerze oraz wiele innych.

**MySQL WorkBench** – Narzędzie wspomagające pracę z bazą danych MySQL. Udostępnia okienkowy interfejs umożliwiający proste i szybkie tworzenie nowych struktur i schematów, wykonywanie zapytań, oraz sprawdzanie wartości zapisanych na bazie.

**Baza danych MySQL** – Darmowa baza danych należąca do Oracle. Posiada bardzo prosty i intuicyjny interfejs. Jest stosunkowo mała i prosta w konfiguracji. Dodatkowo istnieje wiele narzędzi wspomagających pracę z tym systemem bazodanowym co też przyczyniło się do wybrania jej w naszym projekcie.

**Server JBoss** – Darmowy serwer aplikacyjny umożliwiający wdrążanie aplikacji typu Java Enterprise. Posiada wbudowane biblioteki umożliwiające korzystanie z podstawowych technologii, takich jak EJB, Resteasy oraz Log4j. Jest prosty w konfiguracji i wymaga minimalnej konfiguracji przed uruchomieniem. Dodatkowo posiada szeroką społeczność użytkowników, dzięki czemu stosunkowo łatwo znaleźć rozwiązania problemów.

**Gradle** – Narzędzie służące do automatyzacji budowania projektów napisanych w języku Java. Pozwala w łatwy sposób zarządza zależnościami w projekcie, a także zajmuje się za nas dostarczeniem zewnętrznych bibliotek. Daje nam ono możliwość zbudowania projektu jedną komendą. Dodatkowo można wykorzystać inne zadanie by stworzyć wykonywalnego JAR’a z którego możemy uruchomić aplikację poza IDE.

**Git** – System kontroli wersji dzięki któremu mogliśmy w łatwy i szybki sposób dzielić się efektami naszej pracy. Dodatkowo możliwość stworzenia osobnych branchy dla każdego z programistów, ułatwiała pracę i zmniejszała liczbę konfliktów, gdy kilka osób jednocześnie modyfikuje ten sam plik.

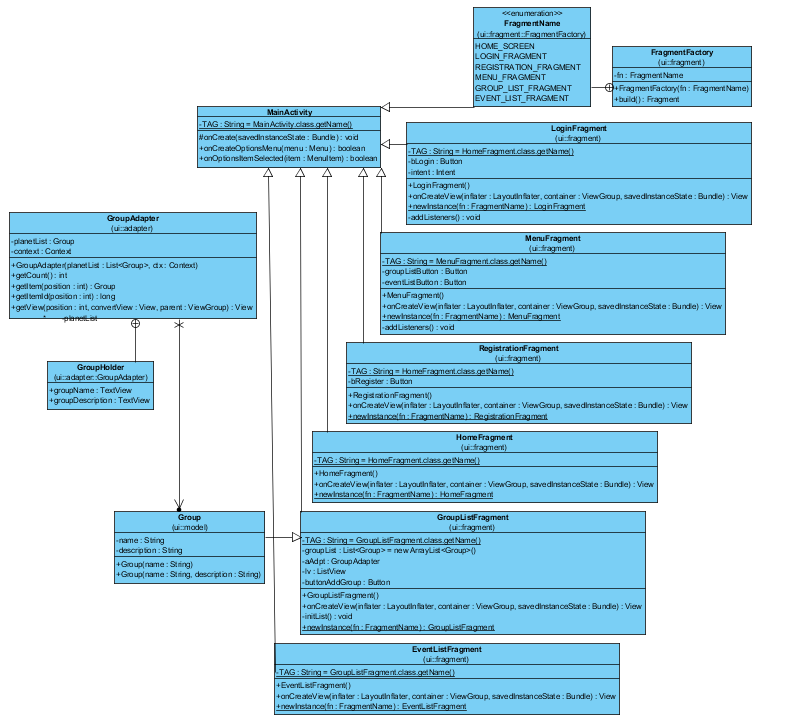
**JUnit** – Framework wspomagający pisanie testów. Został przez nas wybrany ze względu na łatwość użycia. Zostały w nim napisane testy usług po stronie backedn’u.

**Log4j** – Biblioteka ułatwiająca logowanie w aplikacji. Używana w systemie backend’owym w celu zapisania informacji na temat wołanych usług oraz występujących w systemie błędów.

**EJB** – Enterpirse Java Beans. Framework umożliwiający tworzenie ziarenek, odpowiadających za logikę biznesową systemu. Odpowiadają one za komunikację z bazą i weryfikowanie wprowadzanych do systemu informacji.

**Resteasy** – Biblioteka wspomagająca tworzenie REST’owych serwisów dedykowana dla serwera aplikacyjnego JBoss. Została wybrana ze względu na prostą integrację z ziarenkami EJB. Dodatkowo jest domyślnie zainstalowana na serwerze JBoss.

1. **Diagram klas aplikacji mobilnej**



Aplikacje w systemie Android wymagają tzw. Activity czyli Aktywności. Activity jest to jeden z podstawowych komponentów systemu Android. Klasa ta (a raczej jej podklasy) odpowiedzialna jest za interakcję z użytkownikiem, tworzenie okna naszej aplikacji i uruchamianie innych podstawowych komponentów systemowych.

W naszej aplikacji wszystkie z klas zależą od klasy MainActivity, klasy w obrębie której następować będzie cała aktywność aplikacji. Każe z okien aplikacji tworzone jest jako fragment, co znacznie przyspiesza działanie aplikacji, gdyż w obrębie jednego Activity.

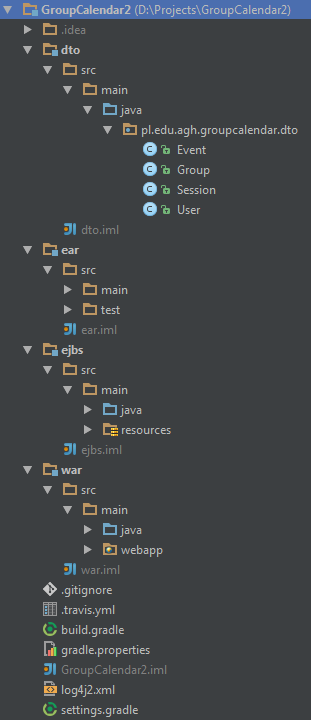
W androidzie można tworzyć dynamiczne interfejsy poprzez łączenie w moduły komponentów UI z aktywnościami. Moduły takie tworzy się za pomocą klasy Fragment, która zachowuje się jak zagnieżdżone aktywności definiujące własne layouty oraz zarządzające własnym cyklem życia.

Kiedy fragment określa własny layout, można skonfigurować go w różne kombinacje z innymi fragmentami wewnątrz aktywności w celu zmodyfikowania konfiguracji layoutu na innych ekranach.

Taka organizacja znacznie przyspiesza działanie aplikacji ponieważ fragment znacznie szybciej odświeża się na ekranie niż utworzenie nowego Activity i ubicie aktualnego.

1. **Struktura aplikacji serwerowej**

Poniżej znajduje się schemat aplikacji wdrążonej na serwerze.

Jak można zauważyć aplikacja ta składa się z 4 modułów.

Pierwszy z nich, *dto*, zawiera klasy obiektów przechowywanych w bazie danych. Dokładniej są to użytkownik, grupa, wydarzenie oraz sesja. Zbudowany jar tego moduły jest dodatkowo eksportowany i wykorzystywany w aplikacji mobilnej, co znacznie ułatwia przepływ informacji między systemami.

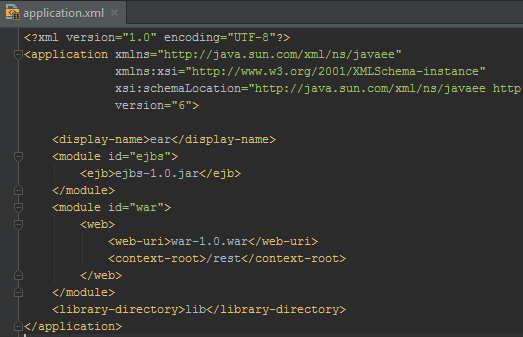
W drugim module, *ejb*s,znajdują sięziarenka EJB, odpowiedzialne za całą logikę aplikacji. Odpowiadają one zarówno za kontakt z bazą danych jak i weryfikację wszystkich obostrzeń co do wywoływanych usług. Moduł ten zawiera 3 bezstanowe ziarenka implementujące 3 lokalne interfejsy. Są one odpowiednio odpowiedzialne za usługi związane z obsługą użytkowników (*AuthBean*), grup (*GroupBean*) oraz wydarzeń (*EventBean*). Dodatkowo w module tym znajduje się też *StartupBean,* odpowiedzialny za konfigurację Log4j przy starcie aplikacji oraz sprawdzenie czy udało się popranie załadować sterownik bazy danych. Z modułu tego powstaje następnie jar.

Kolejnym modułem jest *war* z którego jak można się domyślić powstaje następnie archiwum war. Zawiera on klasy implementujące REST’owe serwisy które są następnie wystawiane na zewnątrz. W aplikacji naszej mamy 3 klasy serwisów (*LoginSerivce*, *GroupService*, *EventService*), odpowiadające analogicznie do ziarenek za usługi związane kolejne z użytkownikami, grupami oraz wydarzeniami. Każdy z serwisów jest dostępny pod inną ścieżką, co ułatwia organizację usług. Kompletna specyfikacja usług znajduje się w dołączonym dokumencie (*Usługi GroupCalendar-x.x*) . W module tym są również zdefiniowane filtry odpowiadające za autentykację użytkowników. Pierwszy z nich *SecurityRequestFilter*, odpowiada za filtrowanie przychodzących zapytań http. Sprawdza on czy wszystkie zawierają w nagłówku klucz aplikacji oraz klucz sesji dla metody innych niż logowanie, tworzenie użytkowników i sprawdzanie statusu aplikacji. Samym generowaniem klucza sesji zajmuje się *AuthBean z* modułu *ejbs.*

Ostatnim modułem w naszej aplikacji jest *ear*. Odpowiada on za zbieranie pozostałych modułów i stworzenie z nim odpowiedniego archiwum ear, które następnie możemy zdeployować na serwerze. Sam w sobie nie zawiera on klas potrzebnych aplikacji. Są w nim natomiast zdefiniowane testy serwisów.

1. **Konfiguracja serwisów**

Konfigurację serwisów i ich ścieżki są definiowane kolejno w kilku miejsach. Pierwszym z nich jest *application.xml*



Definiujemy w nim ścieżki pod którym będziemy mogli odwołać się do usług zdefiniowanych w naszym warze. Kolejnym elementem definiującym ścieżkę jest adnotacja dodana w naszej implementacji interfejsu *Application.* W naszym przypadku nie rozszerzamy w tym miejscu ścieżki.

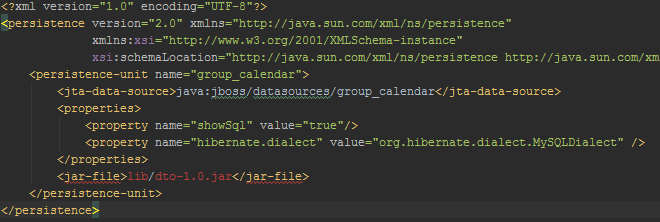


Ostatnim etapem budowania ścieżek pod którymi dostępne są serwisy jest zdefiniowanie ścieżek dla konkretnych serwisów już przy deklaracji konkretnych klas. Robimy to za pomocą odpowiednich adnotacji.



1. **Konfiguracja połączenia z bazą danych**

Za konfigurację połączenia z bazą danych odpowiada wpis w pliku *persistance.xml* znajdującym się w module *ejbs* oraz wpis w pliku *standalone.xml* odpowiadającym za konfigurację serwera.





1. **Logowanie**

Po stronie serwerowej aplikacji jest dodatkowo skonfigurowane logowanie. W tym celu korzystamy z biblioteki Log4j2. Za wczytanie i załadowanie konfiguracji loggera odpowiedzialny jest StartupBean. W odpowiednim pliku konfiguracyjnym definiujemy 2 appendery, tak by jednocześnie wszystkie zdarzenie wypisywały się zarówno w konsoli serwera jak i odpowiednim pliku.