# Rozdział implementacji – backend

## Wzorzec projektowy MVC

Skrót MVC oznacza model – view – controller i jest najpopularniejszym stosowanym wzorcem w tworzeniu oprogramowania. Wyszczególniamy w nim trzy elementy:

* Model który przedstawia logikę biznesową. Wszystkie klasy umieściliśmy w folderze o tej samej nazwie oraz opatrzyliśmy adnotacją @Entity.

Obraz zawierający tekst

Opis wygenerowany automatycznie

* Kontroler używany jest do obsługi żądań użytkowników. Na zdjęciu poniżej widać kontroler z przykładową końcówką dla modelu Facility przedstawionego powyżej.

Obraz zawierający tekst

Opis wygenerowany automatycznie

* Widok czyli interfejs użytkownika przechowujemy w folderze contents dla aplikacji webowej

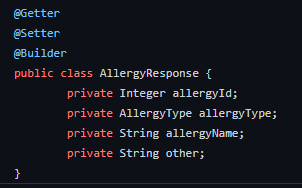
## Wzorzec projektowy factory

Wzorzec ten pozwala na tworzenie wielu obiektów jednego typu, które dziedziczą po jednym interfejsie.

Zastosowaliśmy ten wzorzec dla modeli związanych z wyposażeniem i jego elementami. Szczegóły można zaobserwować na diagramie poniżej. Zdecydowaliśmy się na ten wzorzec ze względu na łatwe dodawanie nowych przedmiotów do wyposażenia, co niewykluczone że wystąpi wraz z rozwojem aplikacji.

## Brak opisu.DTO

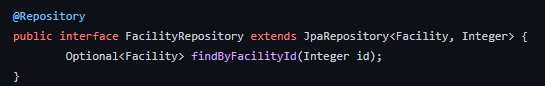
W niniejszej pracy skorzystaliśmy z DTO (Data Transfer Object) jest to jeden z istotniejszych elementów architektury oprogramowania. Pozwala na przenoszenie danych między wieloma warstwami. DTO użyliśmy do niektórych żądań i odpowiedzi w naszym REST API. Dzięki temu treści żądań są krótsze, a odpowiedzi nie zawierają nieskończonych pętli spowodowanych dwukierunkowymi asocjacjami. Na zdjęciu poniżej znajduje się przykład odpowiedzi, a następne zdjęcie przedstawia odpowiadający mu model.



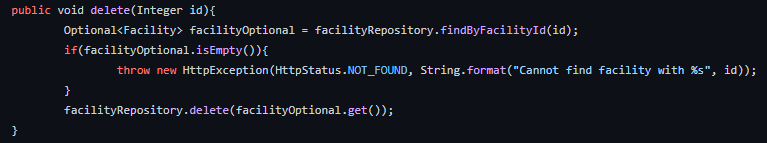


## Komunikacja z bazą danych

Na potrzeby naszej pracy przechowujemy spore zbiory danych powiązanych ze sobą. Do komunikacji z bazą użyliśmy Hibernate wraz z frameworkiem Spring. Aby uzyskać dostęp do danych stworzyliśmy interfejsy DAO (Data Access Object) opatrzone adnotacją @Repository, co można zauważyć na obrazku poniżej.



Na kolejnym obrazku można zobaczyć w klasie serwisowej użycie interfejsu DAO w celu znalezienia rekordu oraz jego usunięcia.



## Dostęp do funkcjonalności wymagających logowania

Z uwagi na używane technologie w niniejszej pracy postanowiliśmy skorzystać ze Spring Security. Pomaga nam w autentykacji oraz uwierzytelnianiu użytkowników. Dodatkowo zastosowaliśmy rolę, które określają zakres uprawnień zalogowanych użytkowników. Co można zobaczyć na poniższym zdjęciu.



## Obsługa błędów z REST API

W niniejszej pracy zostały zaimplementowane mechanizmy zwracające status błędu wraz z wiadomością dla błędnych żądań.

Na obrazku można zobaczyć klasę, którą stosowaliśmy do informowania o błędnych żądaniach.

Obraz zawierający tekst

Opis wygenerowany automatycznie