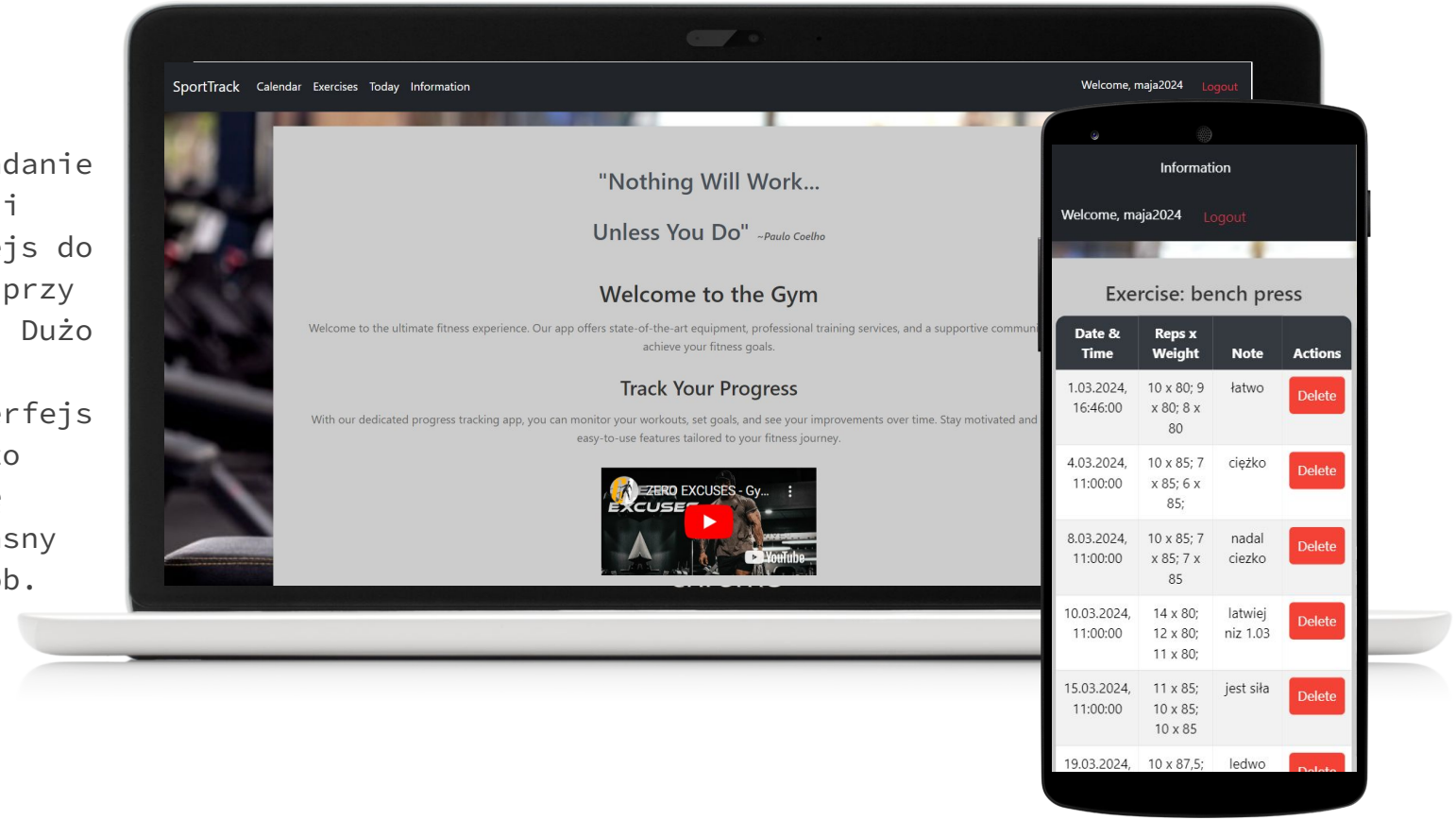
SportTrack

By Marek Szkutnik

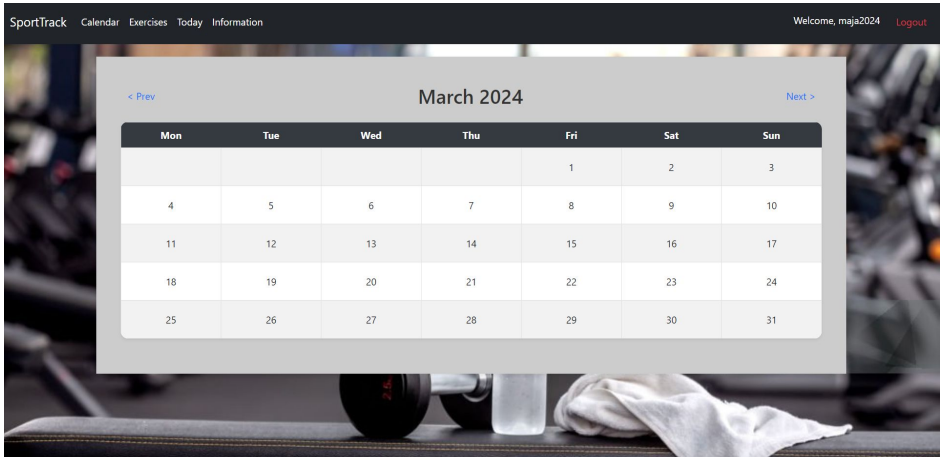
O aplikacji od frontu

SportTrack
aplikacja do śledzenia postępu przy ćwiczeniach

Aplikacja ma za zadanie dostarczać prosty i intuicyjny interfejs do śledzenia postępu przy swoich treningach. Dużo aplikacji ma skomplikowany interfejs i nie zostawia dużo miejsca na swobodę użytkowanie we własny indywidualny sposób.

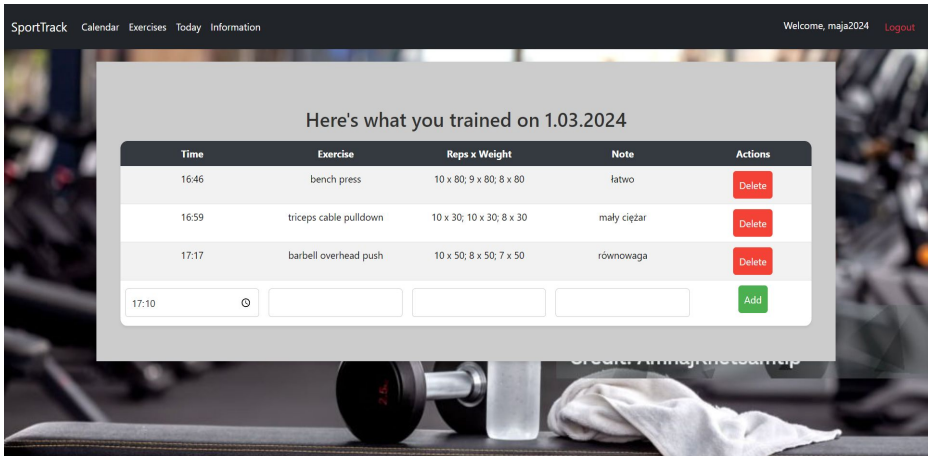


Kalendarz



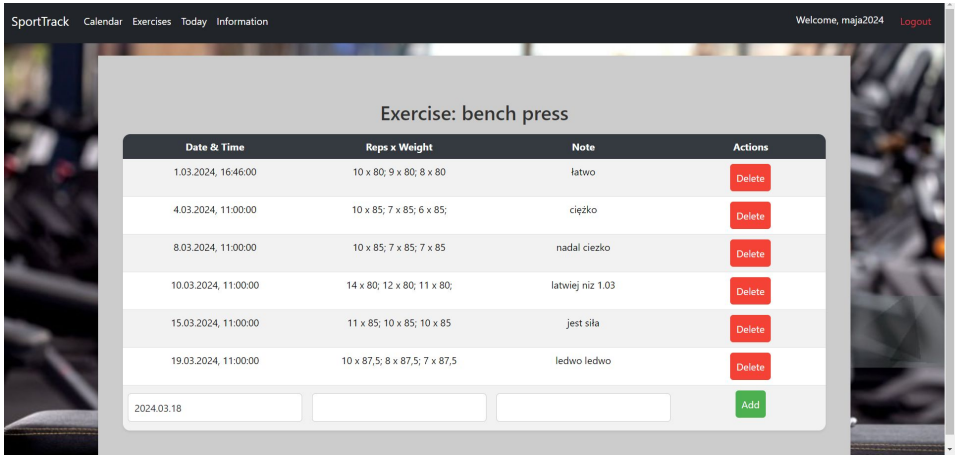
Kalendarz pozwala nam wybrać konkretny dzień w konkretnym miesiącu

Ćwiczenia w danym dniu



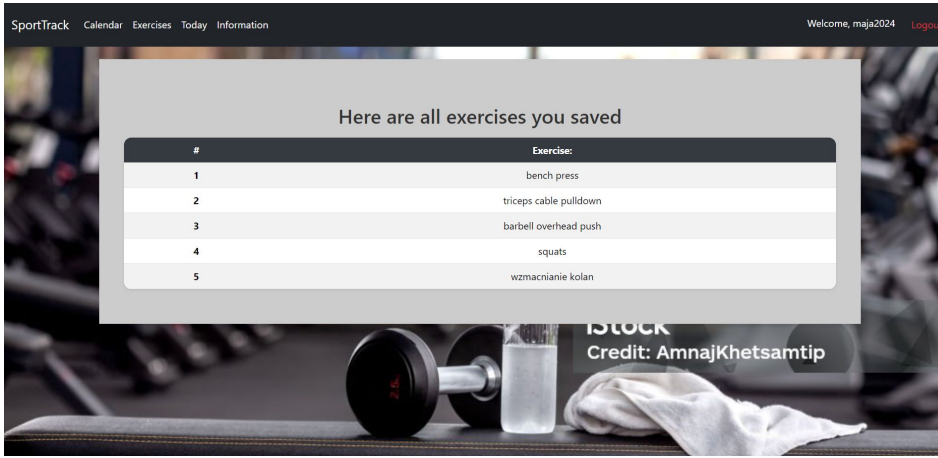
Ta strona pozwala nam przeglądać dodawać oraz usuwać treningi wykonane w konkretnym dniu

Treningi na ćwiczenie



Kolejna strona z tabelą, te pokazuje nam każdy raz gdy wykonywaliśmy dane ćwiczenie. Świetna do śledzenia progresu. Możemy do niej przejść z treningów na dzień albo z wszystkich ćwiczeń

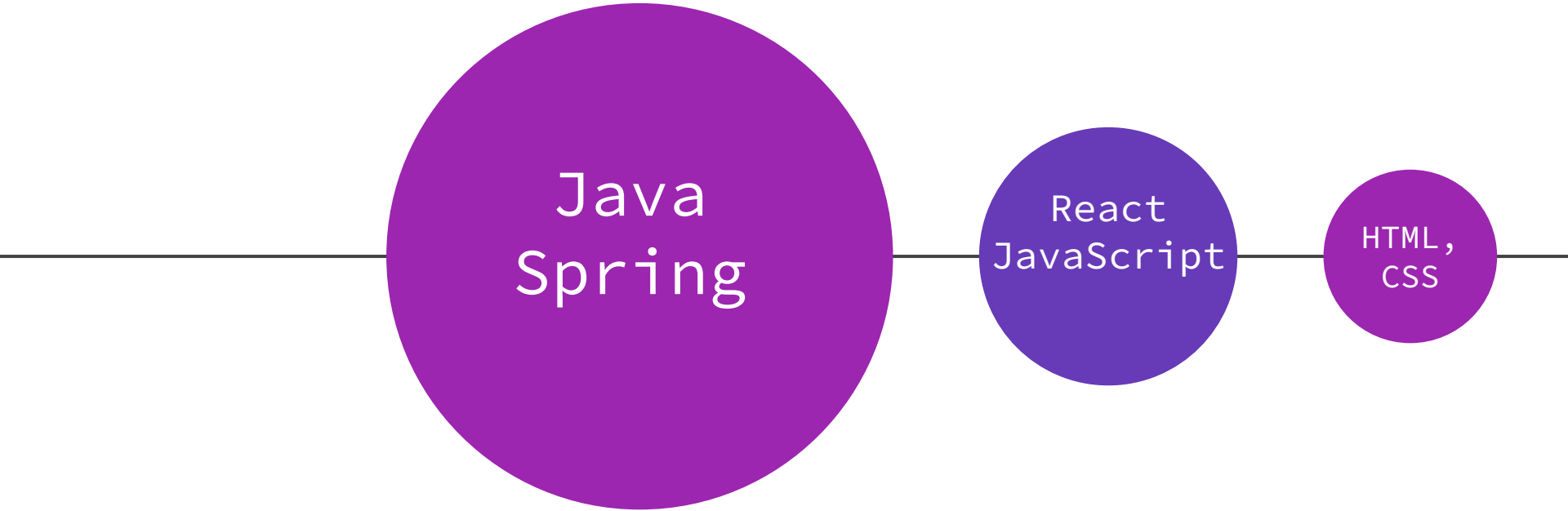
Wszystkie ćwiczenia



Ta strona wyświetla, każde unikalne ćwiczenia jakie sami zapisaliśmy do aplikacji.

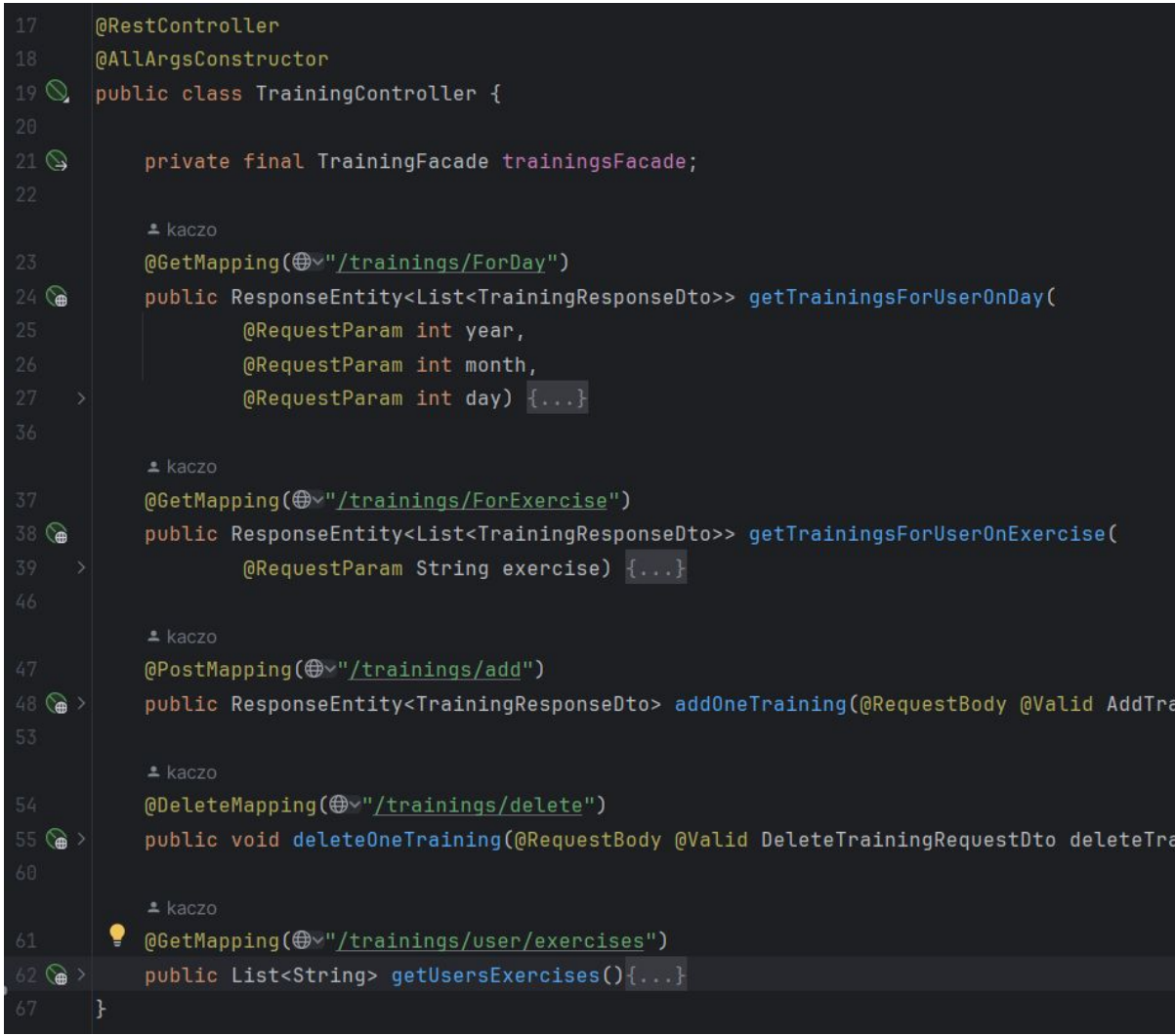
O aplikacji od zaplecza

Technologie

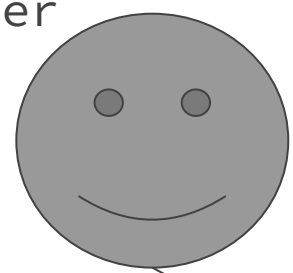


Spring

Aplikacja Springowa wystawia na zewnątrz Api komunikujące się z bazą danych. Obsługuje kilka prostych zapytań związanych z logiką treningów. Możemy je dodawać, usuwać pobierać - wszystkie, w zależności od dnia, w zależności od ćwiczenia.



User



```
GET /trainings/ForDay?year=YYYY&month=MM&day=DD
GET /trainings/ForExercise?exercise=XXXXX
POST /trainings/add
DELETE /trainings/delete
GET @/trainings/user/exercises
```

```
getTrainingsForUserOnDay()
getTrainingsForUserOnExercise()
getDistinctExercisesForUser()
addTraining()
deleteTraining()
```

TrainingFacde

Spring Security

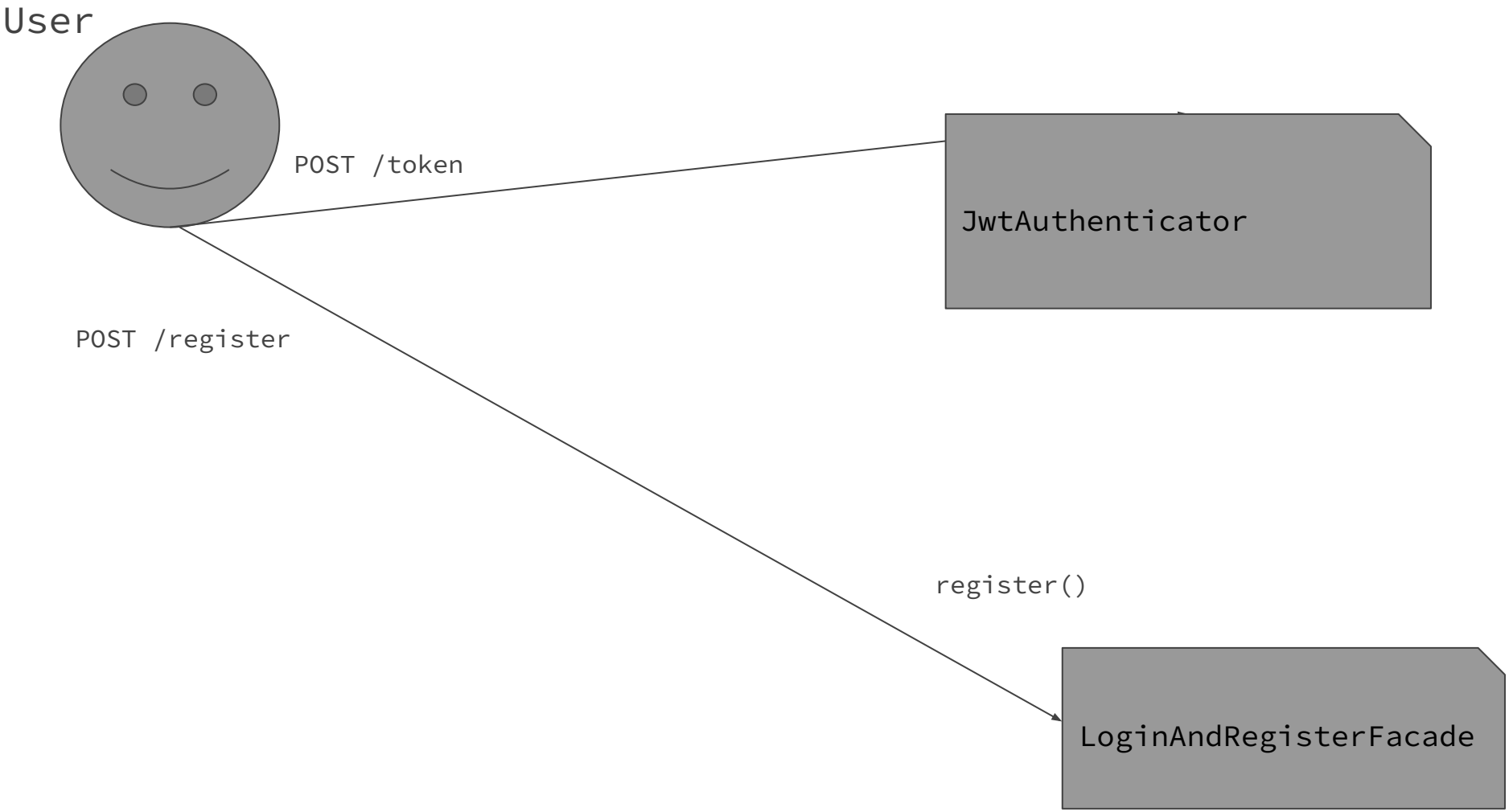
Aplikacja Springowa wystawia również endpointy służące logowaniu oraz rejestracji. Po udanej rejestracji użytkownik może się zalogować co zwróci mu wygenerowany token który następnie będzie służył do weryfikacji użytkownika.

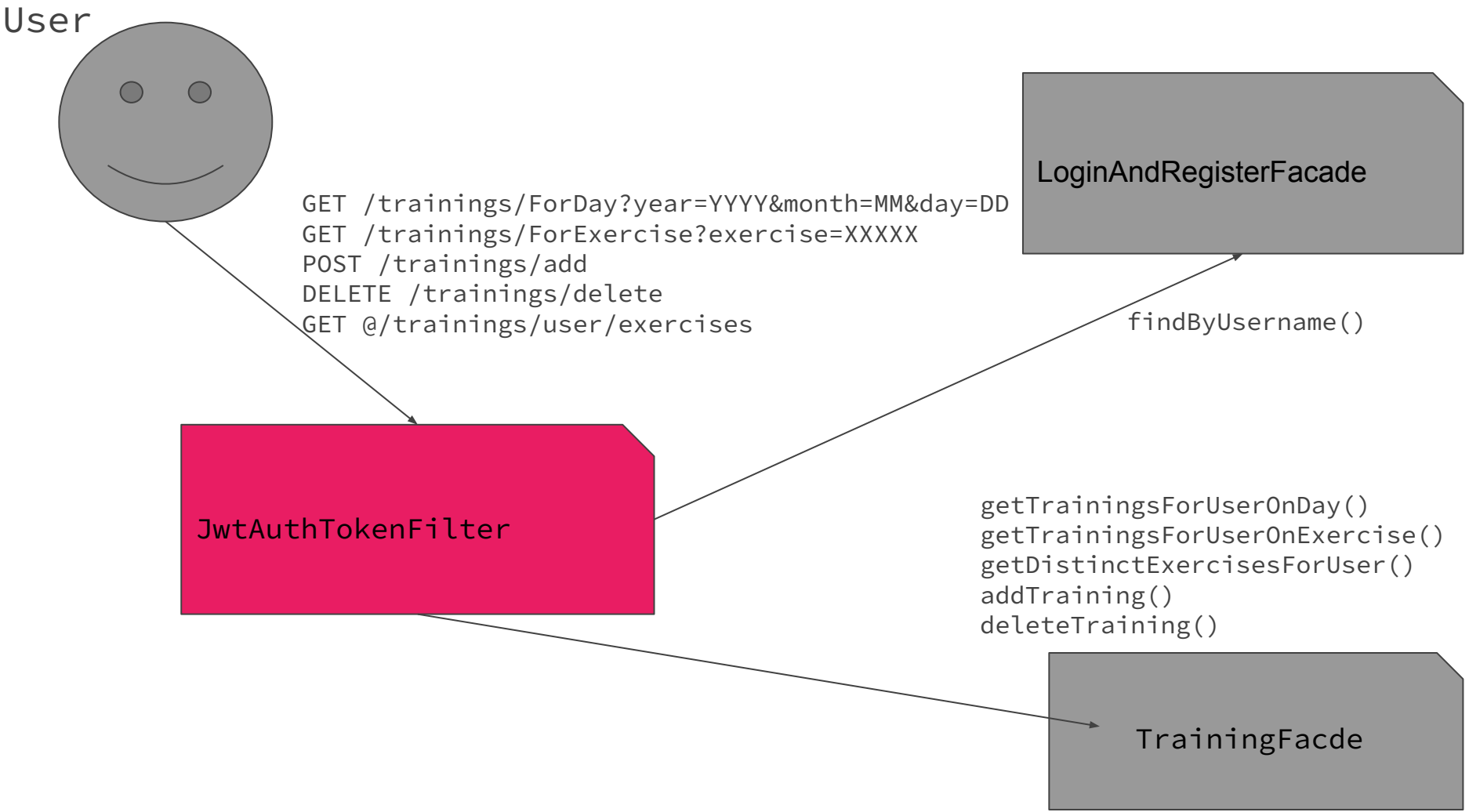
LoginController.java

```java
@RestController
@AllArgsConstructor
public class LoginController {
    JwTAuthenthicator jwTAuthenthicator;
    kaczo
    @PostMapping("/token")
    ResponseEntity<LoginResponseDto> authenthicateAndGenerateToken(@RequestBody @Valid Login
        LoginResponseDto loginResponseDto = jwTAuthenthicator.authenticateAndGenerateToken(t
        return ResponseEntity.ok(loginResponseDto);
    }

}

```

RegisterController.java

```java
@RestController
@AllArgsConstructor
public class RegisterController {
    LoginAndRegisterFacade loginAndRegisterFacade;
    private final PasswordEncoder bCryptPasswordEncoder;
    kaczo
    @PostMapping("/register")
    ResponseEntity<RegisterResponseDto> RegisterUser(@RequestBody @Valid RegisterRequestDt
        RegisterRequestDto userEncodedRequestDto = RegisterRequestDto.builder()
                .password(bCryptPasswordEncoder.encode(userRequestDto.password()))
                .username(userRequestDto.username())
                .build();
        UserResultDto registerResult = loginAndRegisterFacade.register(userEncodedRequestD
        return  ResponseEntity.status(HttpStatus.CREATED).body(new RegisterResponseDto(reg
    }
```


User
POST /token
JwtAuthenticator
POST /register
register()
LoginAndRegisterFacade


User
GET /trainings/ForDay?year=YYYY&month=MM&day=DD
GET /trainings/ForExercise?exercise=XXXXX
POST /trainings/add
DELETE /trainings/delete
GET @/trainings/user/exercises
LoginAndRegisterFacade
findByUsername()
JwtAuthTokenFilter
getTrainingsForUserOnDay()
getTrainingsForUserOnExercise()
getDistinctExercisesForUser()
addTraining()
deleteTraining()
TrainingFacde