# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

# НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ”ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра інформаційних систем та мереж

ЗВІТ

про виконання лабораторної роботи № 3

з дисципліни

« Екстримальне програмування»

Варіант 25

Студент групи КН-311

Прокіпчук О.А.

Прийняв викладач

Щербак С. С.

Львів-2020

**Мета роботи:** здійснити завершення розробки серверної частини проекту та впевнитись у виконанні усіх тестів.

**Хід роботи**

Для забезпечення працездатності ендпоінтів створюю контроллери, що будуть приймати запити та відповідати на них.

@RestController  
**public class** AuthController {  
  
 **private** ParticipantService **participantService**;  
  
 @Autowired  
 **public** AuthController(  
 ParticipantService participantService  
 ) {  
 **this**.**participantService** = participantService;  
 }  
  
 @PostMapping(**"/api/v1/login"**)  
 **public** ResponseEntity<String> login() {  
 **return** ResponseEntity.*ok*(**"Success"**);  
 }  
  
 @PostMapping(**"/api/v1/register"**)  
 **public** ResponseEntity<ParticipantDto> register(@Valid @RequestBody ParticipantDto participantDto) {  
 **return** ResponseEntity  
 .*ok*()  
 .body(**participantService**.register(participantDto));  
 }  
  
}

@RestController  
@RequestMapping(**"api/v1/conference"**)  
**public class** ConferenceController {  
  
 **private** ConferenceService **conferenceService**;  
  
 @Autowired  
 **public** ConferenceController(ConferenceService conferenceService) {  
 **this**.**conferenceService** = conferenceService;  
 }  
  
 @GetMapping  
 **public** ResponseEntity<List<ConferenceDto>> getConferenceList() {  
 **return** ResponseEntity  
 .*ok*()  
 .body(**conferenceService**.getConferenceList());  
 }  
  
 @GetMapping(**"/{conferenceId}"**)  
 **public** ResponseEntity<ConferenceDto> getConference(@PathVariable String conferenceId) {  
 **return** ResponseEntity  
 .*ok*()  
 .body(**conferenceService**.getConference(conferenceId));  
 }  
  
 @PostMapping  
 **public** ResponseEntity<ConferenceDto> createConference(@Valid @RequestBody ConferenceDto conferenceDto) {  
 **return** ResponseEntity  
 .*ok*()  
 .body(**conferenceService**.createConference(conferenceDto));  
 }  
  
 @DeleteMapping(**"/{conferenceId}"**)  
 **public** ResponseEntity<ConferenceDto> cancelConference(@PathVariable String conferenceId) {  
 **return** ResponseEntity  
 .*ok*()  
 .body(**conferenceService**.deleteConference(conferenceId));  
 }  
  
 @PutMapping(**"/add/{conferenceId}"**)  
 **public** ResponseEntity<ConferenceDto> addParticipant(@PathVariable String conferenceId) {  
 **return** ResponseEntity  
 .*ok*()  
 .body(**conferenceService**.addParticipant(conferenceId));  
 }  
  
 @PutMapping(**"/delete/{conferenceId}"**)  
 **public** ResponseEntity<ConferenceDto> removeParticipant(@PathVariable String conferenceId) {  
 **return** ResponseEntity  
 .*ok*()  
 .body(**conferenceService**.removeParticipant(conferenceId));  
 }  
  
 @PutMapping(**"/delete/{conferenceId}/participant/{participantId}"**)  
 **public** ResponseEntity<ConferenceDto> kickParticipant  
 (@PathVariable String conferenceId, @PathVariable String participantId) {  
 **return** ResponseEntity  
 .*ok*()  
 .body(**conferenceService**.kickParticipant(conferenceId, participantId));  
 }  
  
}

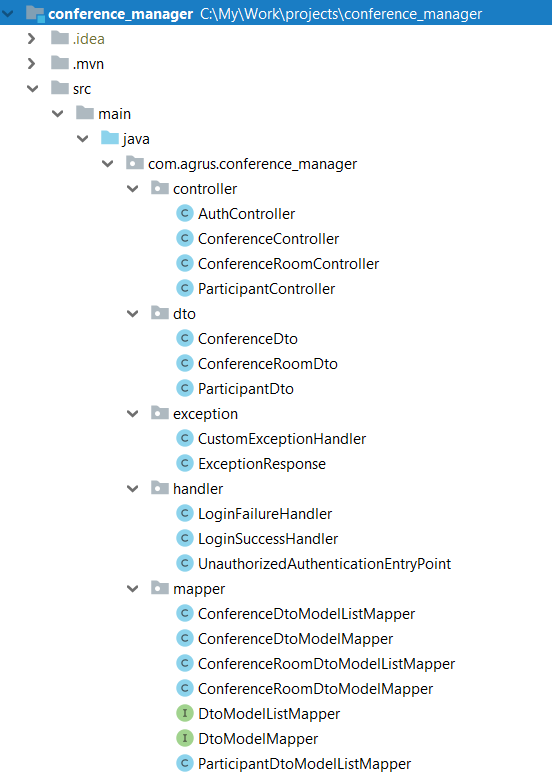
@RestController  
@RequestMapping(**"/api/v1/conference\_room"**)  
**public class** ConferenceRoomController {  
  
 **private** ConferenceRoomService **conferenceRoomService**;  
  
 @Autowired  
 **public** ConferenceRoomController(  
 ConferenceRoomService conferenceRoomService) {  
 **this**.**conferenceRoomService** = conferenceRoomService;  
 }  
  
 @PostMapping  
 **public** ResponseEntity<ConferenceRoomDto> createRoom(@Valid @RequestBody ConferenceRoomDto conferenceRoomDto) {  
 **return** ResponseEntity  
 .*ok*()  
 .body(**conferenceRoomService**.createRoom(conferenceRoomDto));  
 }  
  
 @GetMapping  
 **public** ResponseEntity<List<ConferenceRoomDto>> getRoomList() {  
 **return** ResponseEntity  
 .*ok*()  
 .body(**conferenceRoomService**.getList());  
 }  
  
}

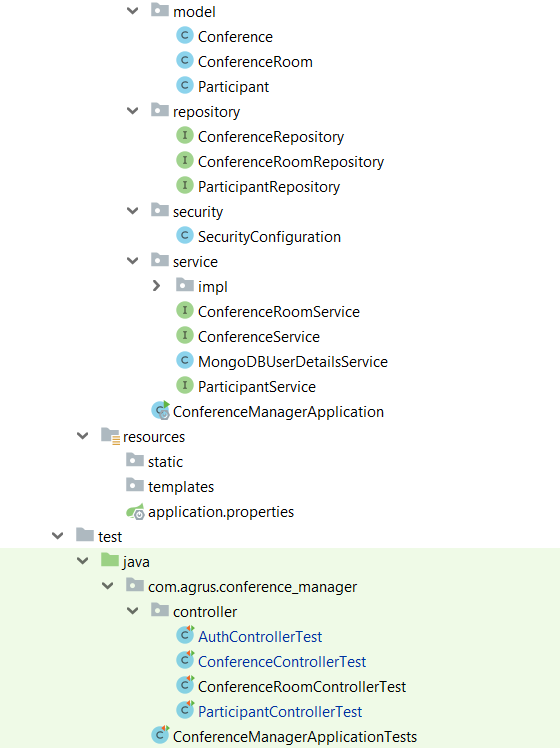
@RestController  
@RequestMapping(**"api/v1/participant"**)  
**public class** ParticipantController {  
  
 **private** ParticipantService **participantService**;  
  
 @Autowired  
 **public** ParticipantController(  
 ParticipantService participantService  
 ) {  
 **this**.**participantService** = participantService;  
 }  
  
 @GetMapping  
 **public** ResponseEntity<ParticipantDto> getCurrentParticipant() {  
 **return** ResponseEntity  
 .*ok*()  
 .body(**participantService**.getCurrentParticipantDto());  
 }  
  
}

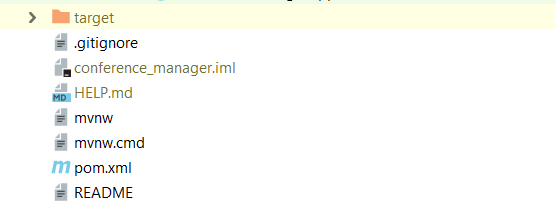
Для роботи контроллерів створюю допоможні класи, а саме:

* Сервіси(для виконання бізнес-логіка)
* Репозиторії(для доступу до бази даних)
* Дто(для запобігання використання сутностей)
* Помилки(для створення та перехоплення помилок)

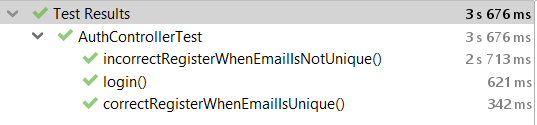
В результаті структура проекту виглядає наступним чином:

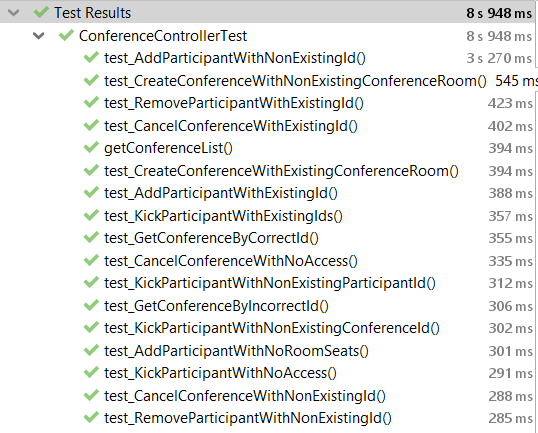


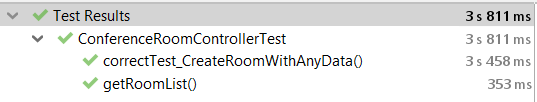


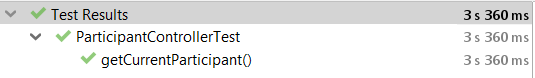


Виконую тести для перевірки виконання усіх вимог









Всі тести успішно виконуються

**Висновок:** на даній лабораторній роботі я здійснив завершення розробки серверної частини проекту та впевнився у виконанні усіх тестів.