**2022년도 졸업프로젝트**

**팀이름: 황제펭귄**

주제: TLS를 구현을 통한 홈페이지 제작 및 보안 채널 로그인 기능 구현

|  |  |  |  |
| --- | --- | --- | --- |
| **작성자** | 이유민 | **작성일자** | 2022.05.26 |

* **금일 진행 상황**

|  |  |
| --- | --- |
| **진행 상황** | **비고** |
| **진행 중**  **완료**   1. **Counserl 기능 Fix**   **ExceptionMessage**  *ERROR\_COUNSEL\_UNAUTHORIZED\_ACCESS*("상담글에 접근할 권한이 없습니다.")  **CounselController**  @DeleteMapping("/counsels/{id}") @ApiOperation(value="상담글 삭제") public BaseResult deleteCounsel(@PathVariable Long id, @RequestParam String loginId) {  try {  counselService.deleteCounsel(id, loginId);  return responseService.successResult();  }catch (Exception e){  return responseService.failResult(  e.getMessage()  );  } }  **CounselCreateReq**  @Data @Builder @RequiredArgsConstructor @AllArgsConstructor public class CounselCreateReq {   @ApiModelProperty(required = true)  private String loginId;  @ApiModelProperty(required = true)  private String title;  @ApiModelProperty(required = true)  private String content;   public Counsel toEntity(User user) {  return Counsel.*builder*()  .title(title)  .content(content)  .createDate(new DateConfig().getDateTime())  .modifyDate(new DateConfig().getDateTime())  .userId(user)  .build();  } }  **CounselRequestRes**  **-loginId,Modifydate 추가**  @Data @Builder @RequiredArgsConstructor @AllArgsConstructor public class CounselRequestRes {   private String loginId;  private String title;  private String content;  private String createDate;  private String modifyDate; }  **CounselUpdateReq**  **-Swagger 추가**  **Counsel**  @Builder @Getter @Entity @NoArgsConstructor @AllArgsConstructor public class Counsel {   @ManyToOne(fetch = FetchType.*LAZY*)  @JoinColumn(name="userId")  private User userId;   @Id  @GeneratedValue(strategy = GenerationType.*IDENTITY*)  private Long id;   @Column(columnDefinition = "text", nullable = false)  private String title;   @Column(columnDefinition = "text", nullable = false)  private String content;   @Column  private String createDate;   @Column  private String modifyDate;   public CounselRequestRes toDto(){  return CounselRequestRes.*builder*()  .loginId(userId.getLoginId())  .title(title)  .content(content)  .createDate(createDate)  .modifyDate(modifyDate)  .build();  } }  **CounselRepository**  **-쿼리문 수정**  @Query("update Counsel c set c.title = ?2, c.content = ?3, c.modifyDate = ?4 where c.id = ?1")  void updateCounsel(@Param("id")Long id, @Param("title") String title, @Param("content") String content, @Param("modifyDate") String modifyDate);  **CounselService**  @Service @RequiredArgsConstructor public class CounselService {   private final CounselRepository counselRepository;  private final UserRepository userRepository;   @Transactional  public void createCounsel(CounselCreateReq counselCreateReq) {  String loginId = counselCreateReq.getLoginId();  // 먼저 해당 사용자가 있는 지 검증  if(!userRepository.existsByLoginId(loginId)){  throw new CounselException(ExceptionMessages.*ERROR\_USER\_NOT\_FOUND*);  }  try {  counselRepository.save(counselCreateReq.toEntity(  userRepository.findByLoginId(loginId).get())  );  } catch (Exception e) {  e.printStackTrace();  throw new CounselException("상담글 생성에 실패했습니다.");  }  }   @Transactional  public List<CounselRequestRes> listAllCounsel() {  return counselRepository.findAll()  .stream()  .map(Counsel::toDto)  .collect(Collectors.toList());  }   @Transactional  public Counsel getSingleCounsel (Long id) {  if(!counselRepository.existsById(id)){  throw new CounselException(ExceptionMessages.*ERROR\_COUNSEL\_NOT\_EXIST*);  }  return counselRepository  .findById(id)  .orElseThrow(() -> new CounselException(ExceptionMessages.*ERROR\_UNDEFINED*));  }   @Transactional  public void updateCounsel(CounselUpdateReq counselUpdateReq) {  if(!counselRepository.existsById(counselUpdateReq.getId())){  throw new CounselException(ExceptionMessages.*ERROR\_EVENT\_NOT\_EXIST*);  }  Counsel counsel = counselRepository.getById(counselUpdateReq.getId());  if(!counsel.getUserId().getLoginId().equals(counselUpdateReq.getLoginId())){  throw new CounselException(ExceptionMessages.*ERROR\_COUNSEL\_UNAUTHORIZED\_ACCESS*);  }  try {  counselRepository.updateCounsel(counselUpdateReq.getId(),  counselUpdateReq.getTitle(),  counselUpdateReq.getContent(),  new DateConfig().getDateTime());  }catch (Exception e){  e.printStackTrace();  throw new CounselException("상담글 업데이트에 실패했습니다.");  }  }   @Transactional  public void deleteCounsel(Long id, String loginId) {  if(!counselRepository.existsById(id)){  throw new CounselException(ExceptionMessages.*ERROR\_COUNSEL\_NOT\_EXIST*);  }  Counsel counsel = counselRepository.getById(id);  if(!counsel.getUserId().getLoginId().equals(loginId)){  throw new CounselException(ExceptionMessages.*ERROR\_COUNSEL\_UNAUTHORIZED\_ACCESS*);  }  try{  counselRepository.deleteById(id);  }catch (Exception e){  e.printStackTrace();  throw new CounselException("상담글 삭제에 실패했습니다.");  }  }  **진행 예정** |  |
| * **특이사항 / 협업 사항** | |