**2022년도 졸업프로젝트[Back-end]**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **작성자** | 이유민 | **작성일자** | 2022.09.20 |

* **금일 진행 상황**

|  |  |
| --- | --- |
| **진행 상황** | **비고** |
| **질문 및 진행방식에 대한 논의**  **진행 중**  **완료**  **1. CreateDate 기준 Sorting 정렬.**  .sorted(Comparator.*comparing*(Event::getCreateDate).reversed())  **2. Swagger 숫자 재정리.**  **3.OTP=>SecurityCard로 전체 변경**  **4.AccountTransaction 알고리즘 수정(보안카드 관련)**  -AccountController  @PostMapping(path = "/transaction/validateaccount") @ApiOperation(value="2. 계좌 이체(계좌 검증)", notes = "") public BaseResult validAccount(  @ApiParam (value = "계좌 이체 객체", required = true)  @RequestBody TransferReq transferReq ){  try {  accountService.validationAccount(transferReq);  transferMap.put(transferReq.getLoginId(),transferReq);  return responseService.successResult();  }catch (Exception e){  return responseService.failResult(  e.getMessage()  );  } }  @GetMapping (path = "/transaction/getsecuritycarddata") @ApiOperation(value="2. 계좌 이체(보안카드 검증 값 전달)", notes = "랜덤넘버 두개를 고릅니다.") public BaseResult sendSecurityCardData(  @ApiParam (value = "사용자 id", required = true)  @RequestParam String loginId ){  try {  SecurityCardRandomRes securityCardRandomRes = securityCardService.selectNumber();  validationSecurityCard.put(loginId,securityCardRandomRes);  return responseService.singleResult(securityCardRandomRes);  }catch (Exception e){  return responseService.failResult(  e.getMessage()  );  } }  @PostMapping (path = "/transaction/checksecuritycard") @ApiOperation(value="2. 계좌 이체(보안카드 검증)", notes = "해시 데이터를 비교합니다.") public BaseResult transaction(  @ApiParam (value = "보안카드 검증 객체", required = true)  @RequestBody SecurityCardValidReq securityCardValidReq  ){  try {  accountService.validSecurityCard(validationSecurityCard.get(securityCardValidReq.getLoginId()), securityCardValidReq);  validationSecurityCard.remove(securityCardValidReq.getLoginId());  return responseService.successResult();  }catch (Exception e){  return responseService.failResult(  e.getMessage()  );  } }  @PostMapping (path = "/transaction/transfer") @ApiOperation(value="2. 계좌 이체", notes = "사용자 아이디와 이체할 계좌, 금액과 해당 계좌의 비밀번호를 받아 이체합니다.") public BaseResult transaction(  @ApiParam (value = "계좌 이체 객체", required = true)  @RequestParam String loginId ){  try {  accountService.transaction(transferMap.get(loginId));  transferMap.remove(loginId);  return responseService.successResult();  }catch (Exception e){  return responseService.failResult(  e.getMessage()  );  } }  -AccountService  @Transactional public void validSecurityCard(SecurityCardRandomRes securityCardRandomRes, SecurityCardValidReq securityCardValidReq) throws NoSuchAlgorithmException {  Member member = memberService.getMember(securityCardValidReq.getLoginId());  securityCardService.validationSecurityCard(member, securityCardRandomRes, securityCardValidReq.getHashedData()); }  public Account getAccount(Long accountNumber){  return accountRepository.findAccountByAccountNumber(accountNumber)  .orElseThrow(() -> new BaseException(ExceptionMessages.*ERROR\_ACCOUNT\_NOT\_FOUND*)); }  -MemberService(회원등록시 보안카드 생성)  Member member = memberRegisterRequest.toEntity(passwordEncoder.encode(password1), Role.*USER*); memberRepository.save(member); securityCardService.createSecurityCard(member);  -SecurityCardRandomRes(랜덤넘버 선택 및 검증에 사용)  @Data @Builder @RequiredArgsConstructor @AllArgsConstructor public class SecurityCardRandomRes {  private int select1;  private int select2;    public boolean checkNumber() {  if (select1 != select2)  return true;  else  return false;  } }  -SecurityCardValidReq(해시 데이터 검증에 사용)  @Data @Builder @RequiredArgsConstructor @AllArgsConstructor public class SecurityCardValidReq {   private String loginId;  private String hashedData; }  -SecurityCard(Entity)  @Builder @Getter @Entity @NoArgsConstructor @AllArgsConstructor  public class SecurityCard {   @Id  private int securityCardPrivateNumber;   @OneToOne(fetch = FetchType.*LAZY*)  @JoinColumn(name="memberId")  private Member memberId;   @Column  private int number1;  private int number2;  private int number3;  private int number4;  private int number5;  private int number6;  private int number7;  private int number8;  private int number9;  private int number10;  private int number11;  private int number12;   public int getSecurityCardNumber(int num) {  switch (num){  case 1:  return this.number1;  case 2:  return this.number2;  case 3:  return this.number3;  case 4:  return this.number4;  case 5:  return this.number5;  case 6:  return this.number6;  case 7:  return this.number7;  case 8:  return this.number8;  case 9:  return this.number9;  case 10:  return this.number10;  case 11:  return this.number11;  case 12:  return this.number12;  }  return -1;  } }  -SecurityCardService(고정된 보안카드 발급,해시 데이터 = refreshtoken+salt)  @Transactional public void createSecurityCard(Member member) {  try {  SecurityCard MemberSecurityCard = SecurityCard.*builder*()  .securityCardPrivateNumber(19741201)  .memberId(member)  .number1(4228)  .number2(6973)  .number3(1011)  .number4(9253)  .number5(1820)  .number6(3563)  .number7(2498)  .number8(1662)  .number9(2890)  .number10(2339)  .number11(1077)  .number12(3840)  .build();  securityCardRepository.save(MemberSecurityCard);  } catch (Exception e) {  e.printStackTrace();  throw new BaseException("보안카드 생성에 실패했습니다.");  } }  @Transactional public void validationSecurityCard(Member member, SecurityCardRandomRes securityCardRandomRes, String hashedData) throws NoSuchAlgorithmException {  SecurityCard securityCard = securityCardRepository.findByMemberId(member)  .orElseThrow(() -> new BaseException(ExceptionMessages.*ERROR\_SECURITYCARD\_NOT\_EXIST*));   // 앞의 2자리  int num1 = securityCard.getSecurityCardNumber(securityCardRandomRes.getSelect1()) / 100;  // 뒤의 2자리  int num2 = securityCard.getSecurityCardNumber(securityCardRandomRes.getSelect2()) % 100;  // 뒤의 4자리  int pk4 = securityCard.getSecurityCardPrivateNumber() % 1000;    String Data = member.getRefreshToken();  String salt = String.*format*("%d%d%d",pk4,num1,num2);  String hashData = hashingData(Data+salt);  if(!hashData.equals(hashedData))  throw new BaseException(ExceptionMessages.*ERROR\_SECURITYCARD\_NOT\_MATCH*); }  public SecurityCardRandomRes selectNumber(){  SecurityCardRandomRes securityCardRandomRes = new SecurityCardRandomRes();  while(true) {  securityCardRandomRes.setSelect1(random.nextInt(12));  securityCardRandomRes.setSelect2(random.nextInt(12));  if(securityCardRandomRes.checkNumber())  break;;  }  return securityCardRandomRes;  }   private String hashingData(String data) throws NoSuchAlgorithmException {  MessageDigest md = MessageDigest.*getInstance*("SHA-256");  md.update(data.getBytes());  return bytesToHex(md.digest());  }   private String bytesToHex(byte[] bytes){  StringBuilder builder = new StringBuilder();  for (byte b : bytes){  builder.append(String.*format*("%02x",b));  }  return builder.toString();  } }  **진행 예정** | re |
| * **특이사항 / 협업 사항** | |