



SEMI-PROJECT



BLOOD CENTER

2팀
전현식, 손한이, 양수경, 윤정서 피동기



INDEX

1. 프로젝트 개요

프로젝트 목표
프로젝트 일정

2. 팀 구성 및 계획

담당 업무 소개
계획 및 일정

3. 수행 절차

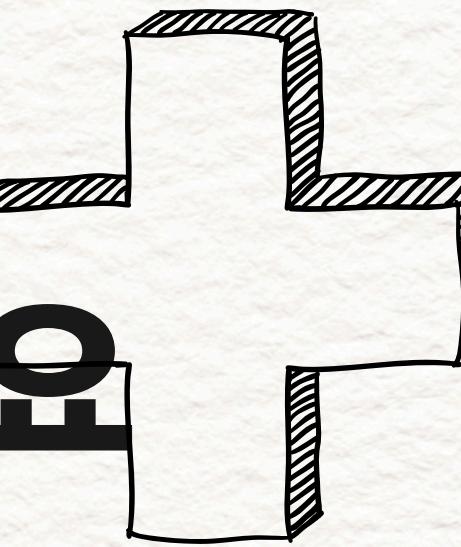
시스템 흐름도
수행 결과
기능 시연

4. 자체 평가 의견

자체 평가
QnA



1. 프로젝트 개요

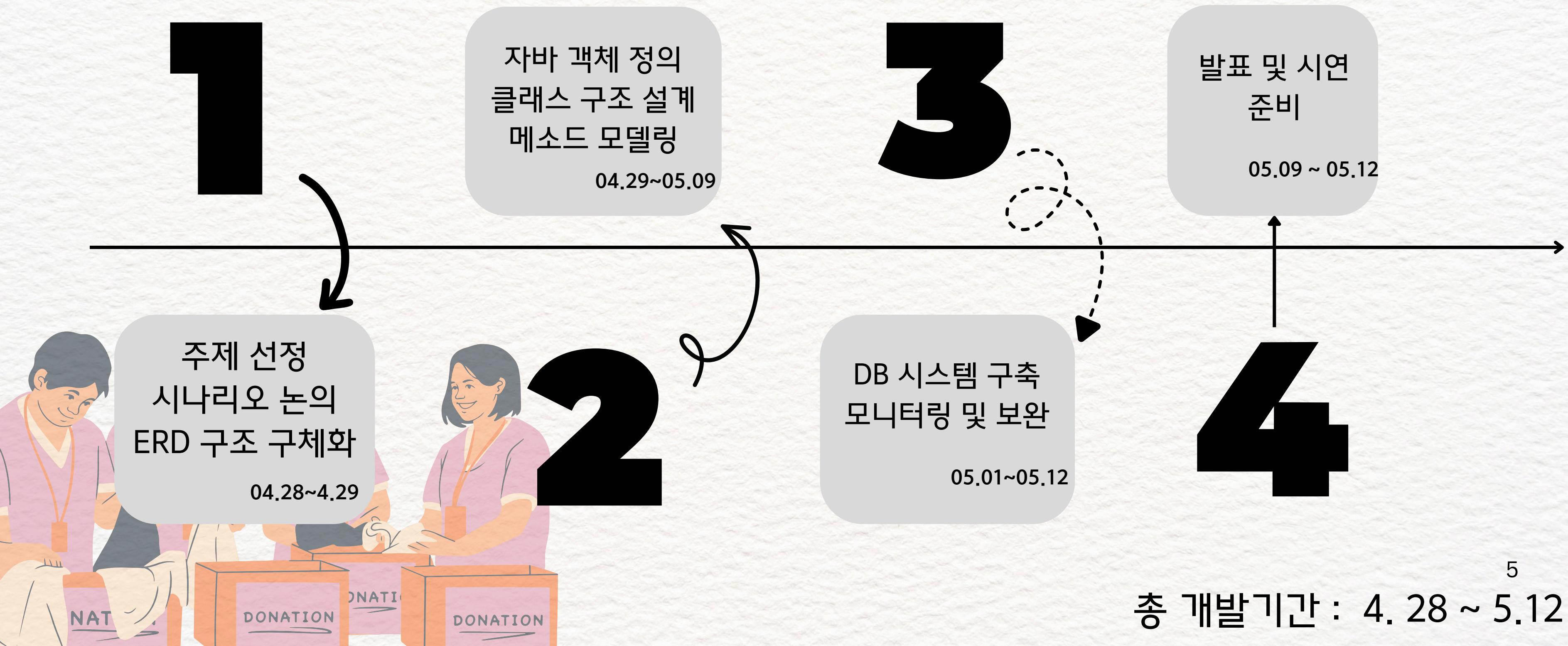


프로젝트 목표

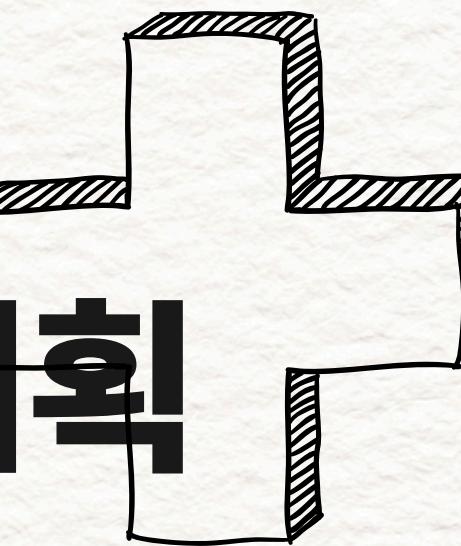
- 대구 지역 헌혈 정보 관리/실행 프로그램 제작
- 레드커넥트 APP / 대한적십자사 웹페이지를 참고
- 사용자 입·출력은 자바 콘솔에서 관리
- (1) 헌혈 희망자 (사용자) 메뉴 / (2) 관리자 메뉴로 분할



PROJECT TIMELINE

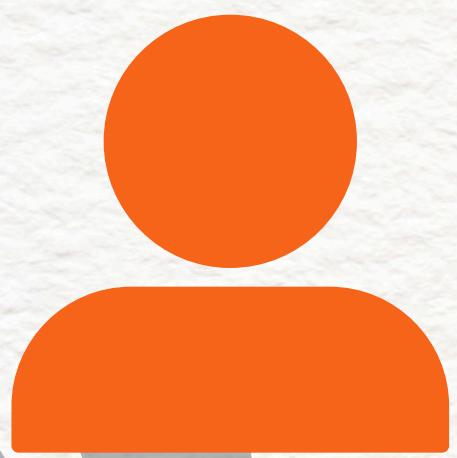


2. 팀 구성 및 계획



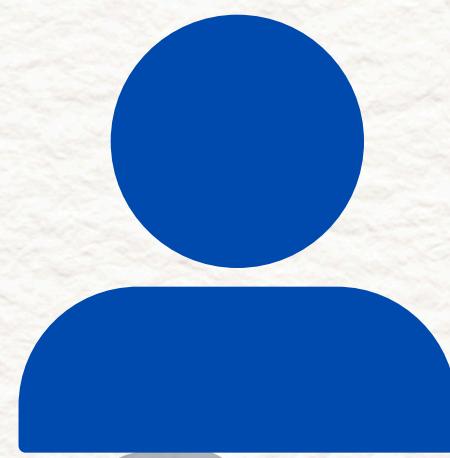
팀원 역할 소개

주제 선정 및 지속적인 논의
테이블 생성 및 데이터 추가



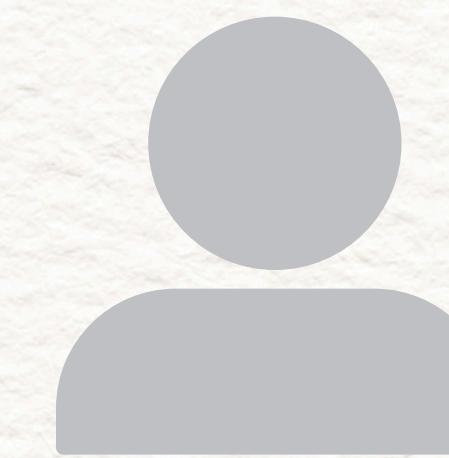
전현식

전반적인 진행 상황
점검 및 동기화
깃허브관리
현재 혈액형별 재고량
예약변경/취소 기능
메소드 담당



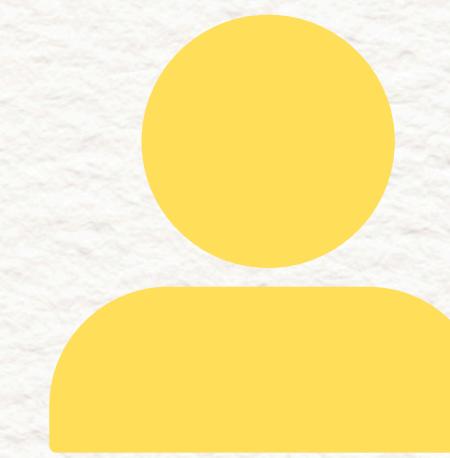
손한이

DB관리
헌혈예약 가능
헌혈 후 혈액재고 추가
기능



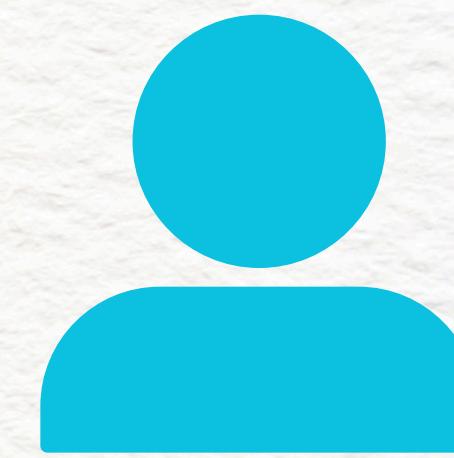
양수경

(헌혈자, 수혈자, 헌혈예약,
혈액정보, 수혈, 수혈한 혈액 표시)
6가지 랜덤생성 메소드 담당
사용 가능 혈액 상세보기 담당
오라클 가상테이블 만들기 담당
기간 만료된 혈액 폐기 sql 담당



윤정서

헌혈내역보기 및
관리자 기능 담당



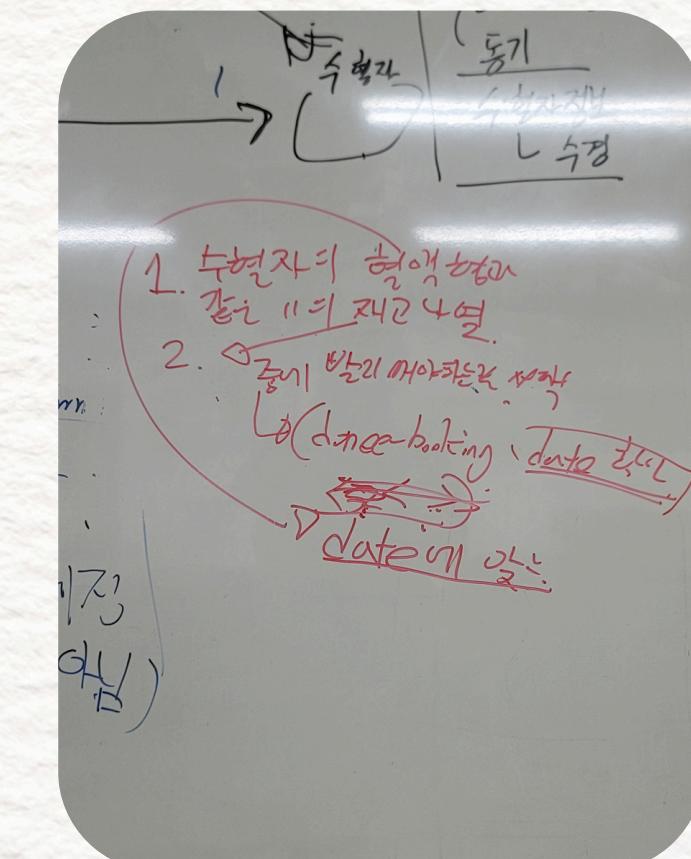
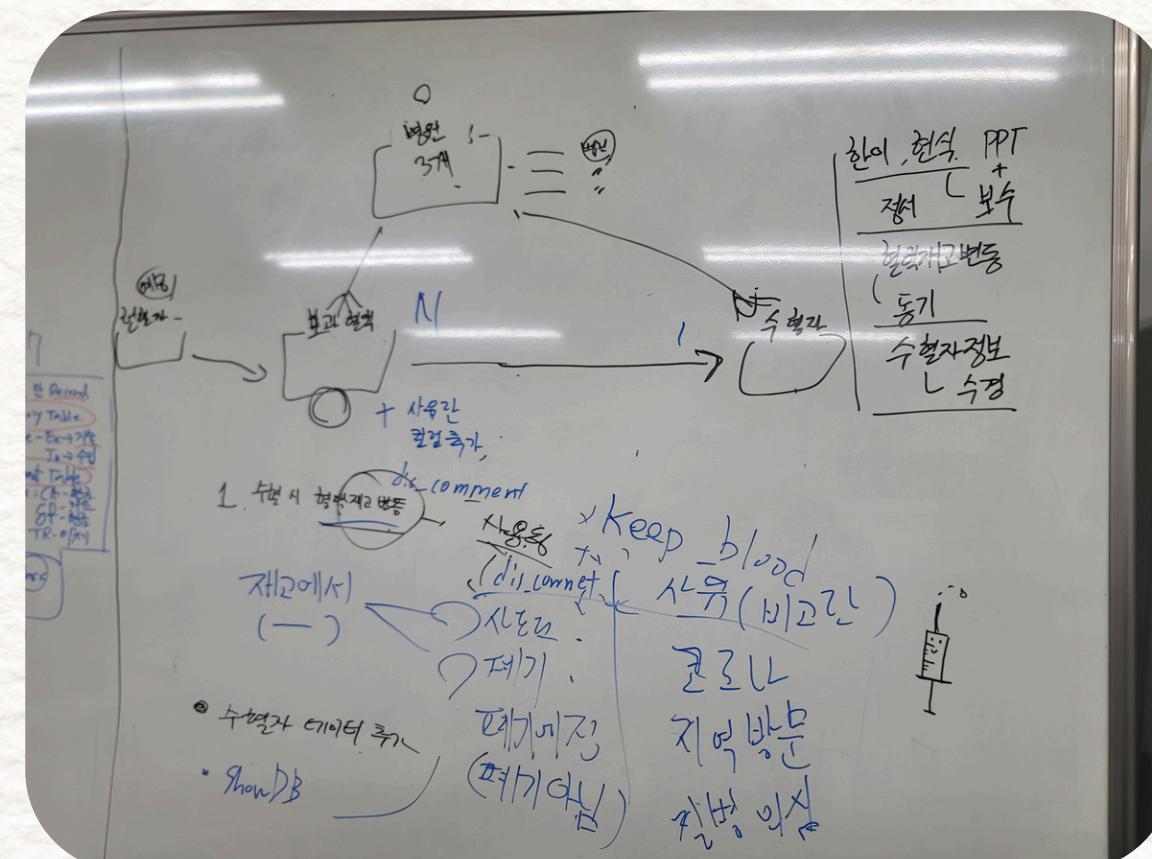
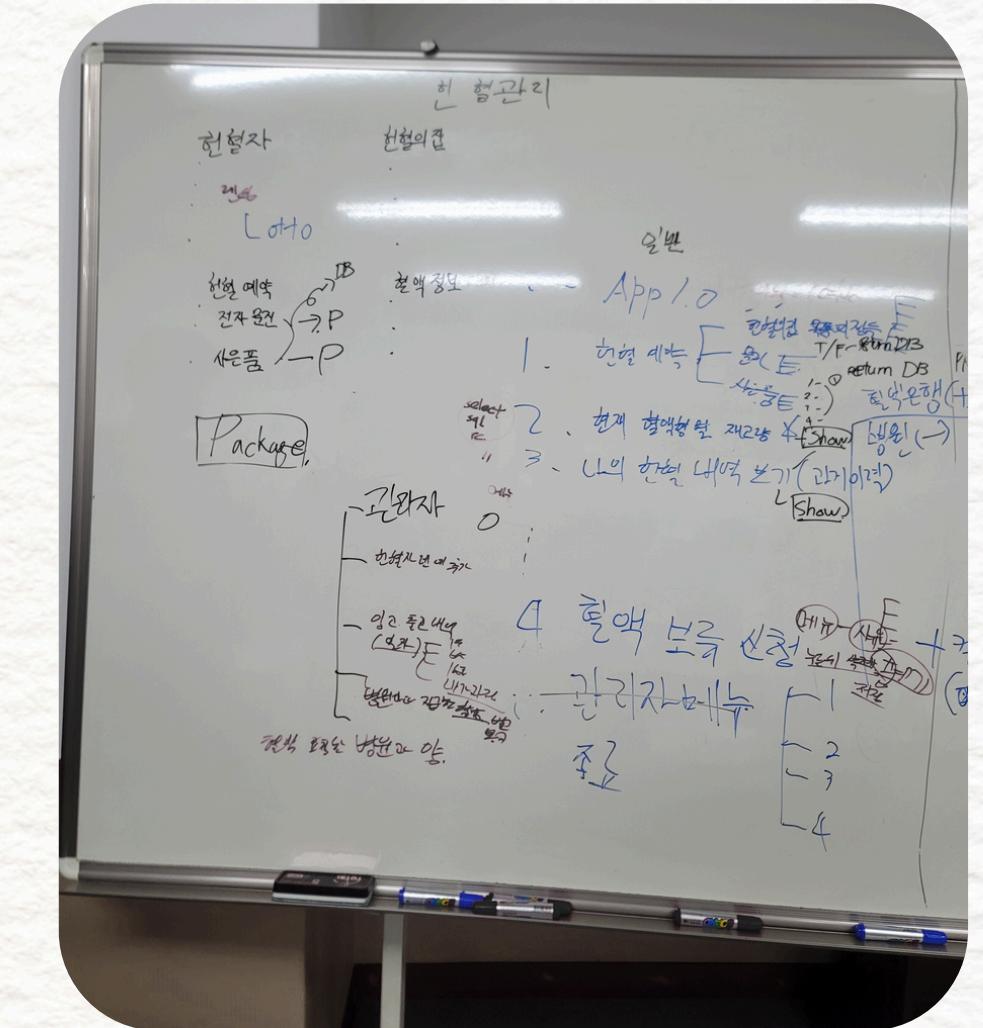
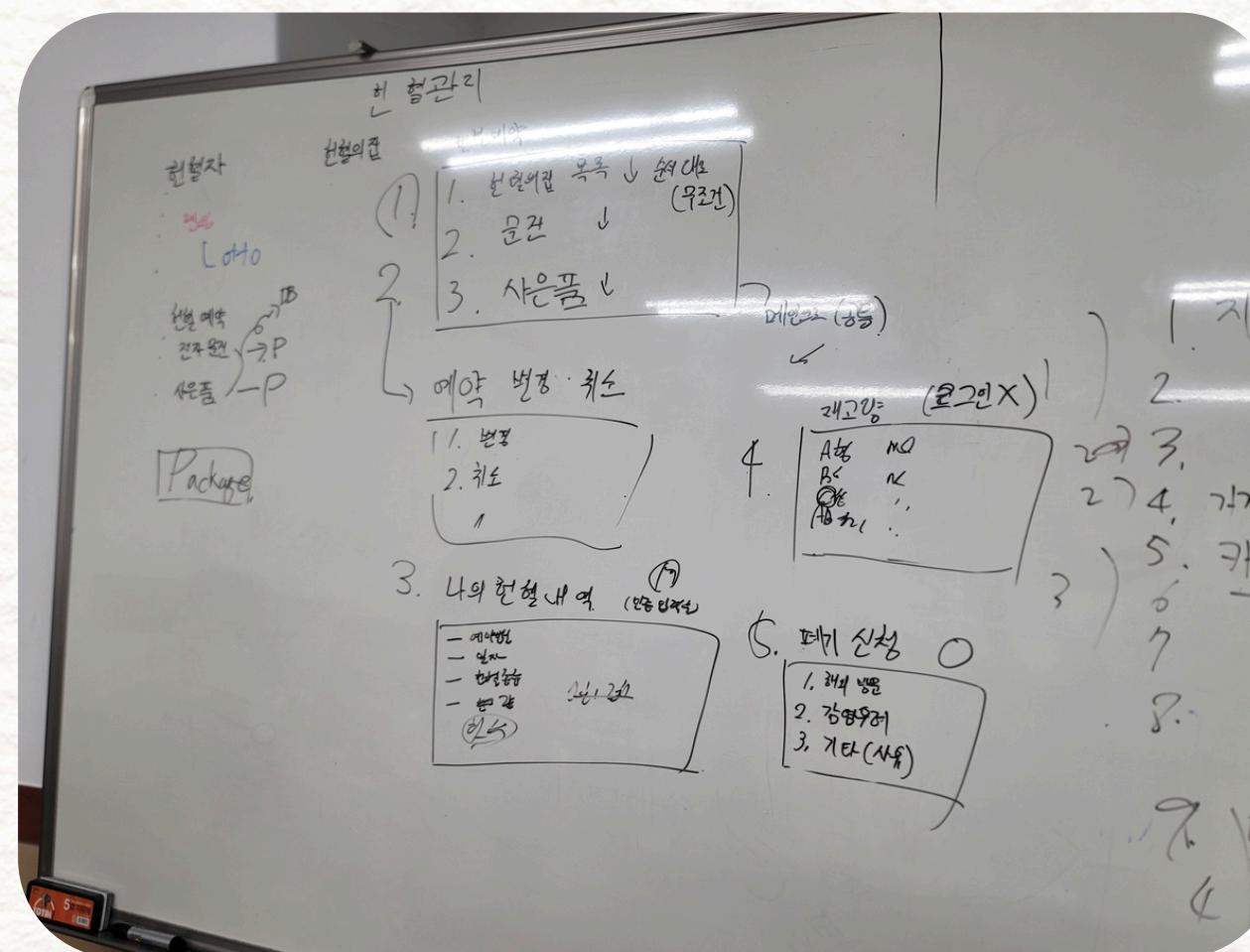
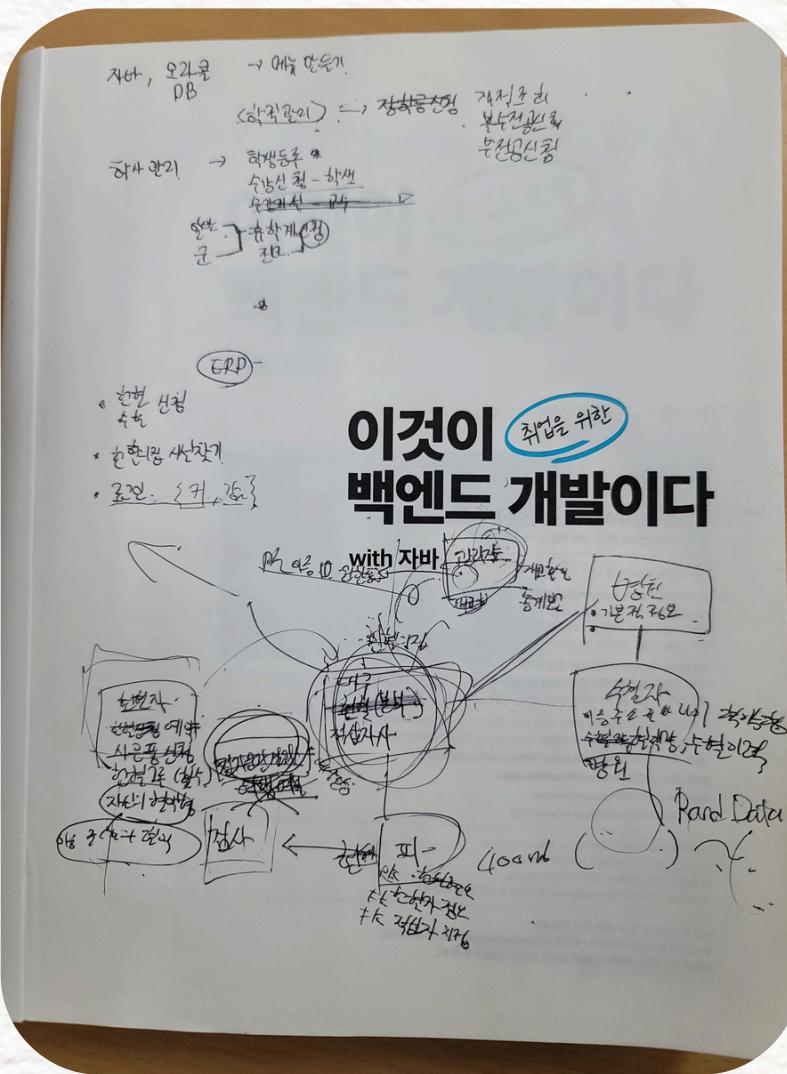
피동기

헌혈완료된 혈액폐기관련
메소드 담당

3. 프로젝트 수행절차



프로젝트 초기구상 및 중간점검회의자료



기술 스택

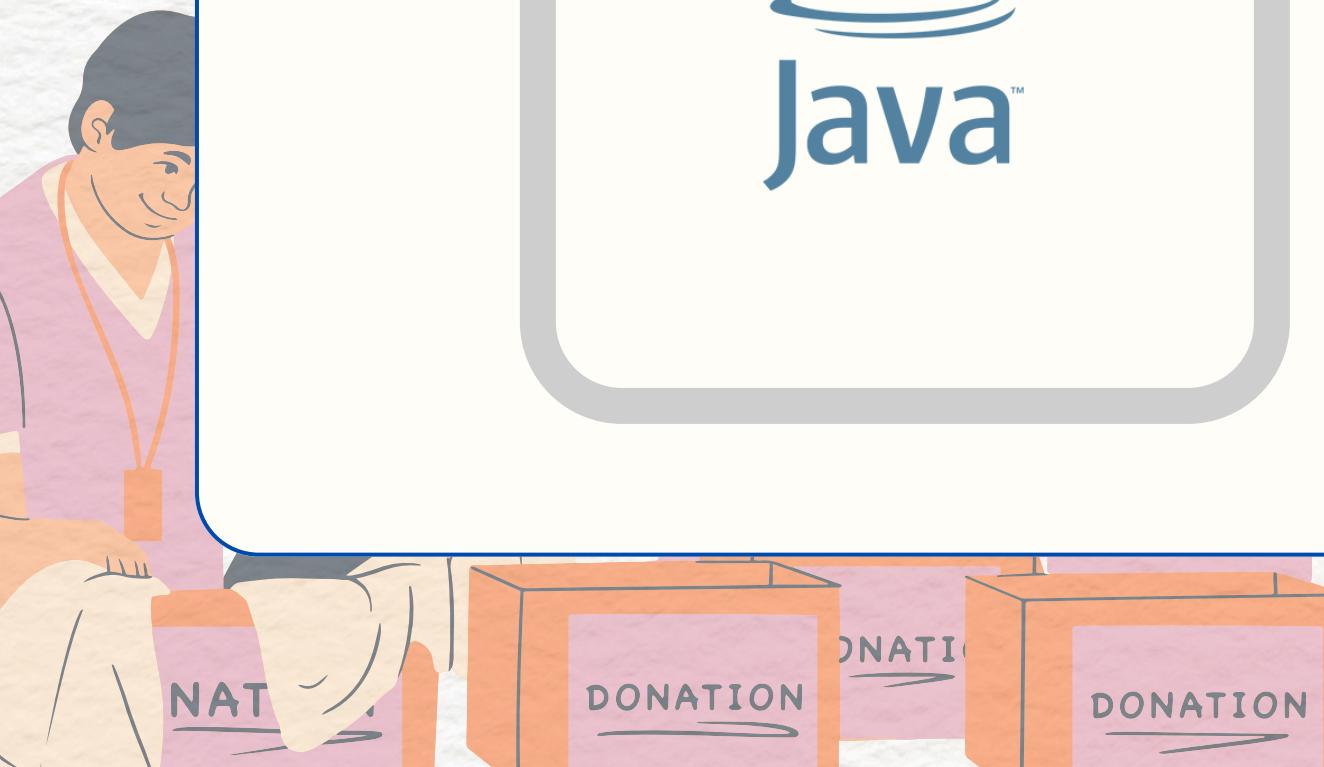
Back End



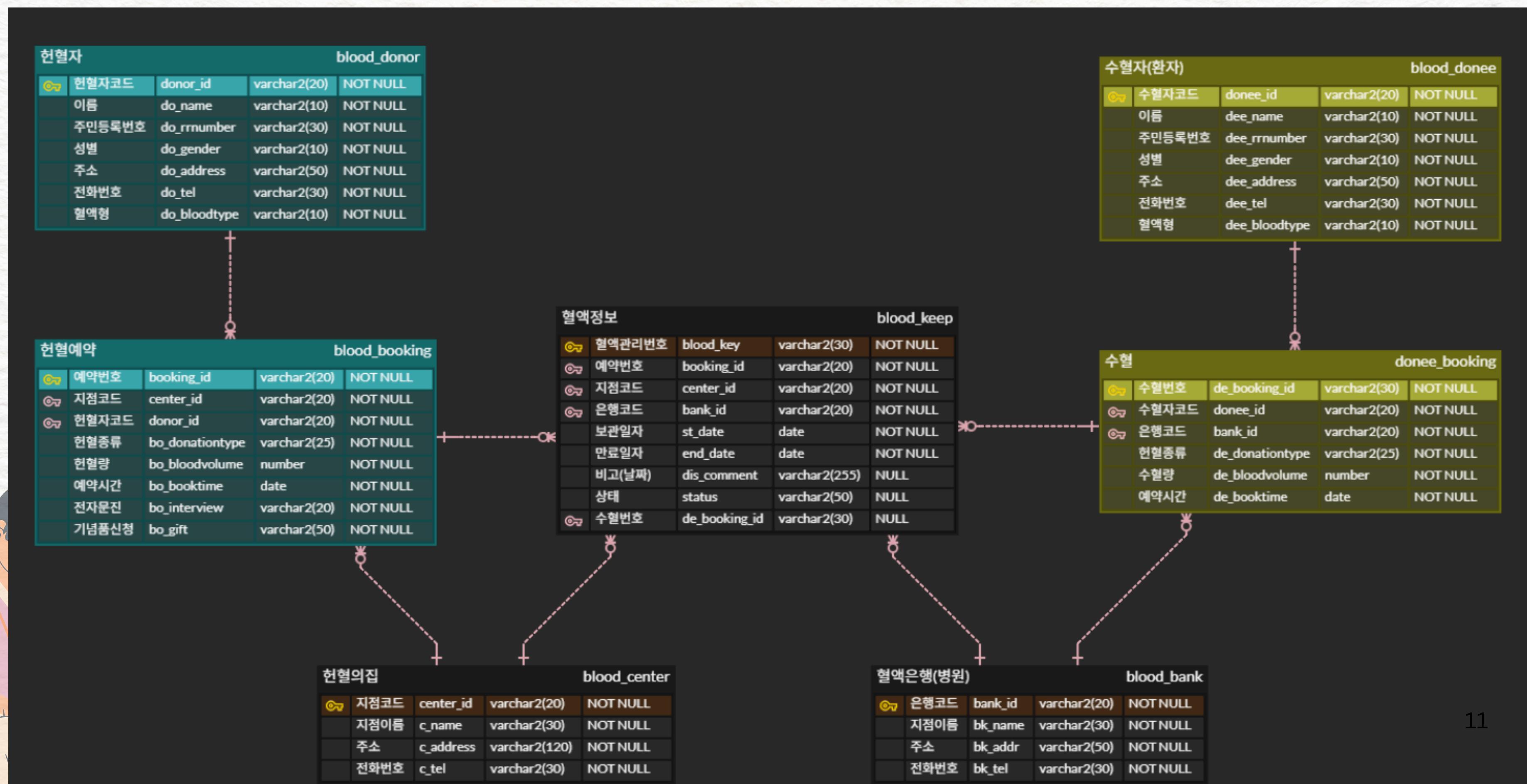
Database



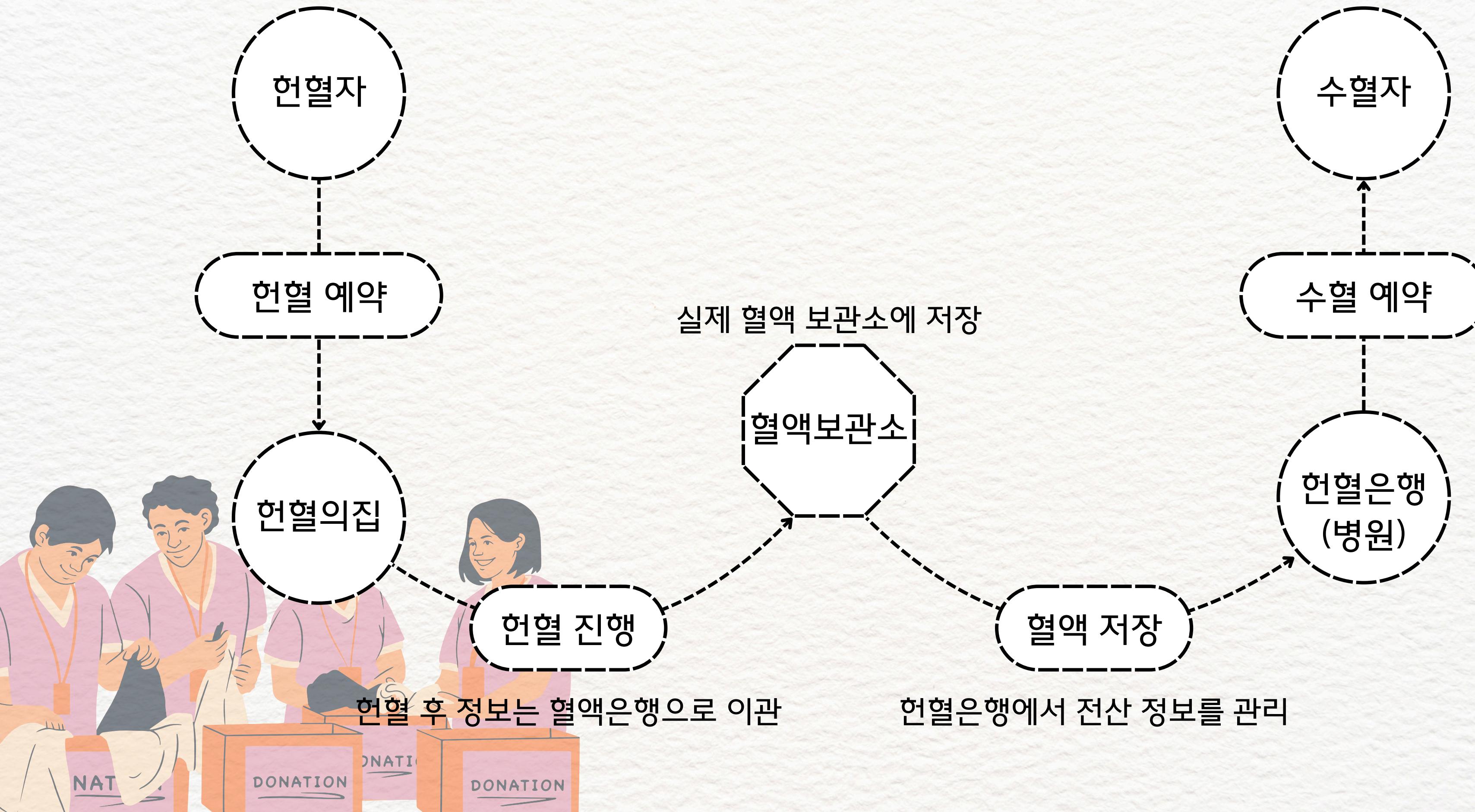
Tools



ERD 클라우드를 이용한 테이블 관계도



시스템 흐름도



참고 자료

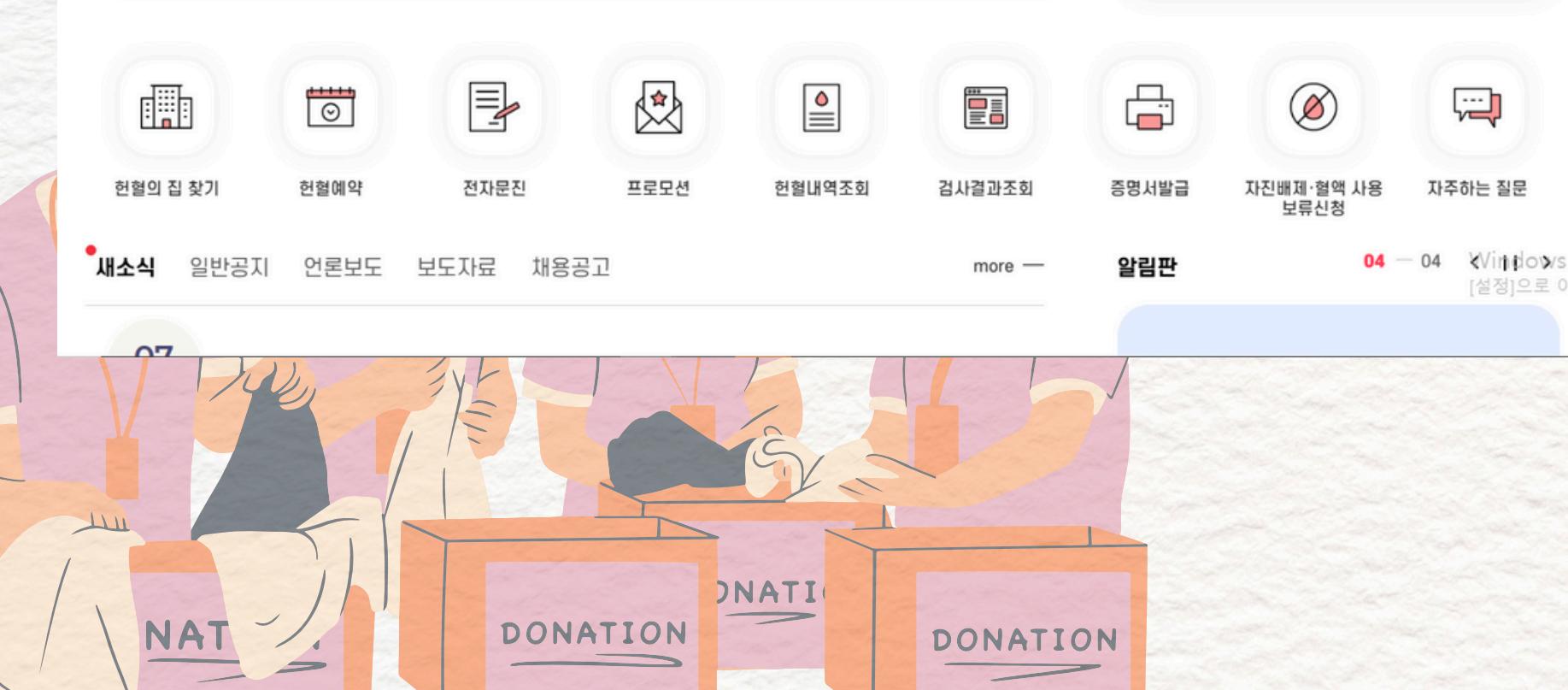
**2025년 3월 4일 이후
영국 유럽 여행/체류 관련 헌혈 제한 기준 변경**

기존 헌혈 제외 대상(25.3.3.까지 해당)
 '80년~'96년까지 1개월 이상 영국 거주
 '97년~ 현재까지 3개월 이상 영국 거주
 '80년~ 현재까지 5년 이상 유럽국 거주
 '80년 이후 영국, 프랑스에서 수혈

변경된 헌혈 제외 대상(25.3.4.부터 적용)
 '80년~'96년까지 3개월 이상 영국 거주
 '80년~'01년까지 5년 이상
 프랑스, 아일랜드 거주
 '80년 이후 영국, 프랑스, 아일랜드에서 수혈

함께 만드는 즐거운 헌혈,
 안전한 수혈, 더 나은 미래

적혈구 혈소판
 5.3 DAY
 오늘의 혈액보유량
 2025.05.08 기준
 전체 5.3일
 A 4.2 B 7.9 O 4.3 AB 4.6



홈 알림판

헌혈의집 찾기 전자문진 공지사항

혈액 전달

혈액 출고
 생명나눔을 실천해주셔서 감사합니다.
 혈액검사를 거쳐 안전하다고 판정된 혈액은
 생명을 살리는 데 소중하게 사용될 예정입니다.

혈액 보유 현황

2025년 05월 08일 00:00 기준 혈액 보유 현황

여러분의 헌혈이
 소중한 생명을 살리고 있습니다.

혈액형	보유일수
O	4.3일
A	4.2일
B	7.9일
AB	4.6일

오늘도 8,724명이 헌혈에 동참하고 있습니다.

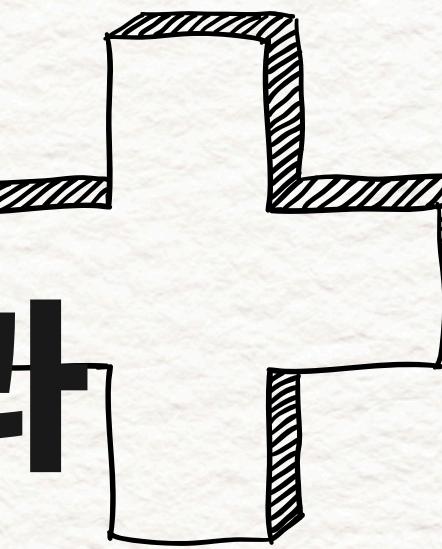
홈 예약 검사결과 더보기

카테고리별 자료 수집:

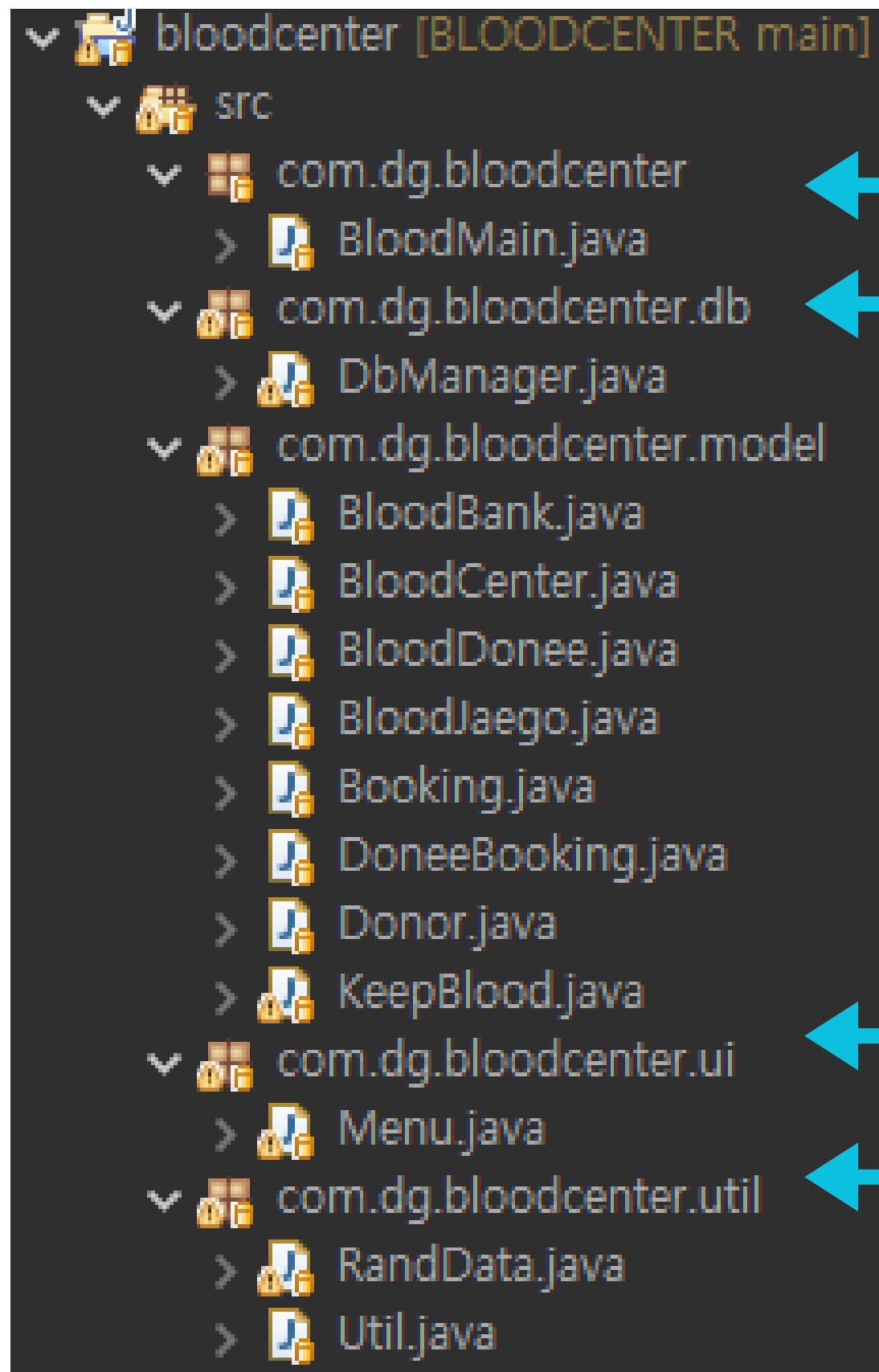
대한적십자사
 혈액관리본부 웹사이트
 및 레드커넥트

수업 때 진행한 은행 앱 개발
 자료 참고

프로젝트 결과



구성



메인

DB 관리 / 메소드 모음

모델링

Getter / Setter 설정

기본 생성자 설정

사용자 메뉴 / 관리자 메뉴

공용 유틸 클래스

(랜덤 데이터, 현재 시각, Y/N 구분 등)

사용자 UI – 1. 헌혈 예약

헌혈 관리 App v1.0.0

- 1. 헌혈 예약
- 2. 예약 변경/취소
- 3. 나의 헌혈 내역 보기
- 4. 현재 혈액형별 재고량
- 5. 혈액 폐기 신청
- 6. 관리자 메뉴
- 7. 종료

메뉴 선택: 1

id를 입력해주세요 admin

pw를 입력해주세요 1234

====관리자 메뉴=====

- 1. 사용자 데이터 추가
- 2. 사용자 예약 데이터 추가
- 3. 혈액형별 헌혈자 검색
- 4. 헌혈예약 정보 전체검색
- 5. 기간이 만료된 혈액 폐기
- 6. 수혈
- 7. 이전 메뉴로 돌아가기

항목 선택:

메뉴 선택: 1
사용자 ID 입력(ex DN-1): dasd
※ 존재하지 않는 사용자 ID입니다. 다시 입력해주세요.

사용자 ID 입력(ex DN-1): dn-33
※ 존재하지 않는 사용자 ID입니다. 다시 입력해주세요.
사용자 ID 입력(ex DN-1): dn-1
※ 존재하지 않는 사용자 ID입니다. 다시 입력해주세요.

사용자 ID 입력(ex DN-1): DN-1

9

지점코드: CT09

지점명: 경북대북문센터

주소: 대구 북구 대학로 83 (산격동 1341-2) 경북대학교

연락처: 053-421-6235

=====

예약할 지점 번호를 선택하세요 (예: 1~9): 213

잘못된 인력입니다. 다시 선택해주세요.

예약할 지점 번호를 선택하세요 (예: 1~9): 9

1. 사용자 ID 입력 메소드 실행

2. 유효한 사용자 ID 검사

3. 올바른 입력시 대구 헌혈의 집 리스트 출력

4. 헌혈을 실행할 헌혈의 집 선택

전자문진 실행

```
public String askValidDonorId(Scanner s, DbManager db) {  
    while (true) {  
        System.out.print("사용자 ID 입력(ex DN-1): ");  
        String donorId = s.next();  
        if (db.findDonorId(donorId)) {  
            return donorId;  
        } else {  
            System.out.println("※ 존재하지 않는 사용자 ID입니다. 다시 입력해주세요.");  
        }  
    }  
}
```

사용자 UI - 1. 헌혈 예약

헌혈 관리 App v1.0.0

- 1. 헌혈 예약**
- 2. 예약 변경/취소
- 3. 나의 헌혈 내역 보기
- 4. 현재 혈액형별 재고량
- 5. 혈액 폐기 신청
- 6. 관리자 메뉴
- 7. 종료

메뉴 선택: 6

id를 입력해주세요 admin

pw를 입력해주세요 1234

====관리자 메뉴=====

- 1. 사용자 데이터 추가
- 2. 사용자 예약 데이터 추가
- 3. 혈액형별 헌혈자 검색
- 4. 헌혈예약 정보 전체검색
- 5. 기간이 만료된 혈액 폐기
- 6. 수혈
- 7. 이전 메뉴로 돌아가기

항목 선택:

전자문진을 시작합니다!

최근 1개월 이내에 해외여행을 다녀온 적이 있습니까? (y/n): n

현재 복용 중인 약물이 있습니까? (y/n): n

최근 1주일 내 감기 증상이 있었습니까? (y/n): n

현재 체온이 37.5도 이상입니까? (y/n): n

전자문진 결과: 헌혈가능

현재 체온이 37.5도 이상입니까? (y/n): y

전자문진 결과: 헌혈불가

* 헌혈 불가로 기념품 선택이 취소됩니다.

예약 정보가 추가되었습니다!

* 헌혈이 불가능한 사용자입니다. 혈액 정보가 추가되지 않습니다.

헌혈 기념품을 선택해주세요!

- 1. 커피교환권(5000원)
- 2. 영화관람권
- 3. 손톱깎이세트
- 4. 기념품을 선택하지 않겠습니다.

기념품 선택: 3

예약 정보가 추가되었습니다!

혈액 정보가 추가되었습니다!

4. 간략한 전자문진을 실행 (Y/N)

문진 중 1개라도 Y → 헌혈불가

<헌혈가능> 사용자는 예약기록·혈액재고 데이터 생성

<헌혈불가> 사용자는 예약기록 데이터만 생성

5. <헌혈가능> 사용자는 헌혈 기념품 선택메뉴로 이동

BOOKING_ID	CENTER_ID	DONOR_ID	BO_DONATIONTYPE	BO_BLOODVOLUME	BO_BOOKTIME	BO_INTERVIEW	BO_GIFT
BO-34	CT09	DN-1	혈소판 성분헌혈	250	25/05/12	가능	손톱깎이세트

예약정보

BLOOD_KEY	BOOKING_ID	CENTER_ID	BANK_ID	ST_DATE	END_DATE	DIS_COMMENT	STATUS	DE_BOOKING_ID
B_KEY-34	BO-34	CT09	BK01	25/05/12	25/06/16	(null)	(null)	(null)

사용자 UI – 1. 헌혈 예약

```
public String interviewMenu(Scanner s) {  
    System.out.println("전자문진을 시작합니다!");  
    System.out.println();  
    String travel = Util.askYesNo(s, "최근 1개월 이내에 해외여행을 다녀온 적이 있습니까? (y/n)");  
    String medicine = Util.askYesNo(s, "현재 복용 중인 약물이 있습니까? (y/n): ");  
    String cold = Util.askYesNo(s, "최근 1주일 내 감기 증상이 있었습니까? (y/n): ");  
    String fever = Util.askYesNo(s, "현재 체온이 37.5도 이상입니까? (y/n): ");  
    boolean isEligible =  
        travel.equals("n") &&  
        medicine.equals("n") &&  
        cold.equals("n") &&  
        fever.equals("n");  
    String result;  
    if (isEligible) {  
        result = "가능";  
    } else {  
        result = "불가";  
    }  
  
    // 헌혈자 ID 검증 메소드  
    public boolean findDonorId(String donorId) {  
        PreparedStatement pstmt = null;  
        String sql = "select count(*) from blood_donor where donor_id = ?";  
        if (rs.next()) {  
            return rs.getInt(1) > 0;  
        }  
        return false;  
    }  
}
```

Util 클래스의 askYesNo 메소드를 활용
y/n 답변 중 하나라도 y가 존재한다면
헌혈 불가

```
public static String askYesNo(Scanner s, String question) {  
    while (true) {  
        System.out.print(question);  
        String input = s.nextLine().trim().toLowerCase();  
        if (input.equals("y") || input.equals("n")) {  
            return input;  
        } else {  
            System.out.println("잘못된 입력입니다. 'y' 또는 'n'으로 응답해주세요.");  
        }  
    }  
}
```

헌혈자 ID 검증은 boolean 속성을 사용
sql문의 count(*)로 하나의 행을 반환
true (존재하는 사용자ID) 리턴 혹은
false (존재하지 않는 사용자 ID) 리턴

사용자 UI - 2. 현혈 예약 변경

```
// 현혈 예약 변경 기능
public void modifyDonation(String bookingId, String newDonationType, int newBloodVolume, String newBookDate,
    String newGift) {
    PreparedStatement pstmt = null;
    // 네 테이블 구조(blood_booking)에 맞춰서 SQL 쿼리 컬럼 이름 수정!
    // booking_id, bo_donationtype, bo_bloodvolume, bo_booktime, bo_gift
    String sql = "UPDATE blood_booking SET bo_donationtype = ?, bo_bloodvolume = ?, bo_booktime = to_date(?, 'YY/MM/DD'), bo_gift = ? WHERE booking_id = ?";
    // 컬럼
    // 이름
    // 수정!

    try {
        pstmt = con.prepareStatement(sql);
        pstmt.setString(1, newDonationType); // 새로운 현혈 종류 -> bo_donationtype 컬럼에
        pstmt.setInt(2, newBloodVolume); // 새로운 혈액량 -> bo_bloodvolume 컬럼에
        pstmt.setString(3, newBookDate); // 새로운 예약일자 -> bo_booktime 컬럼에 (to_date로 변환)
        pstmt.setString(4, newGift); // 새로운 사은품 -> bo_gift 컬럼에
        pstmt.setString(5, bookingId); // 변경할 예약 ID -> booking_id 컬럼에

        int rowsAffected = pstmt.executeUpdate(); // SQL 실행

        if (rowsAffected > 0) {
            System.out.println(bookingId + " 예약이 성공적으로 변경되었습니다!");
            con.commit(); // 성공하면 저장!
        } else {
            // 해당 ID의 예약이 없을 경우
            System.out.println("예약 ID " + bookingId + "를 찾을 수 없습니다. 변경할 예약이 없어요.");
            // 이때는 둘백할 필요 없어.
        }
    } catch (SQLException e) {
        showErr(e); // 오류 출력
        try {
            con.rollback(); // 실패하면 되돌리기!
            System.out.println(bookingId + " 예약 변경 중 오류가 발생...");
        } catch (SQLException e1) {
            e1.printStackTrace();
            System.out.println("변경 실패 후 둘백 중에도 오류 발생");
        }
    } finally {
        if (pstmt != null) {
            try {
                pstmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        // Connection은 여기서 닫으면 안 돼! DbManager에서 관리!
    }
}
```

1. 예약자 ID 입력

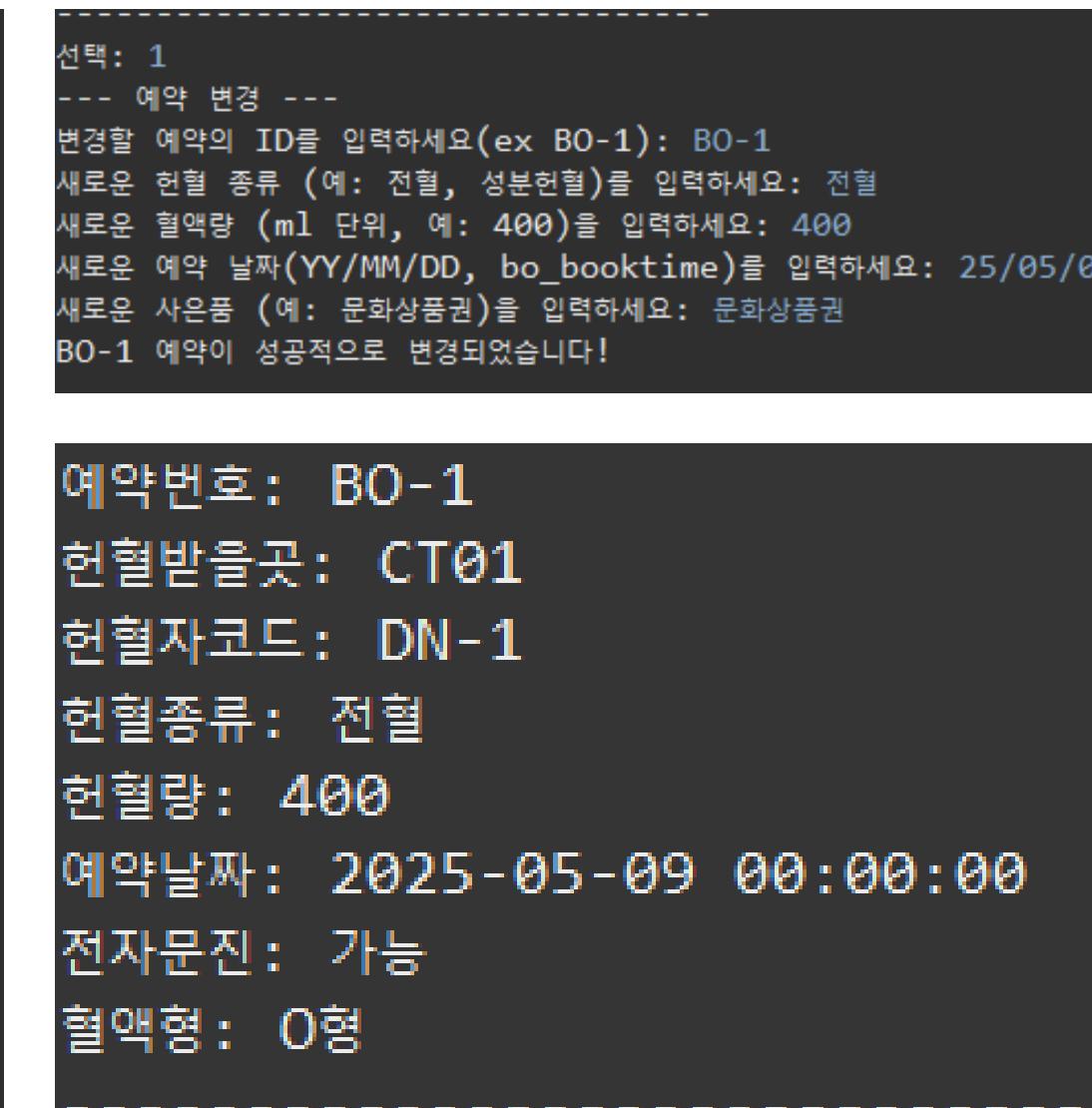
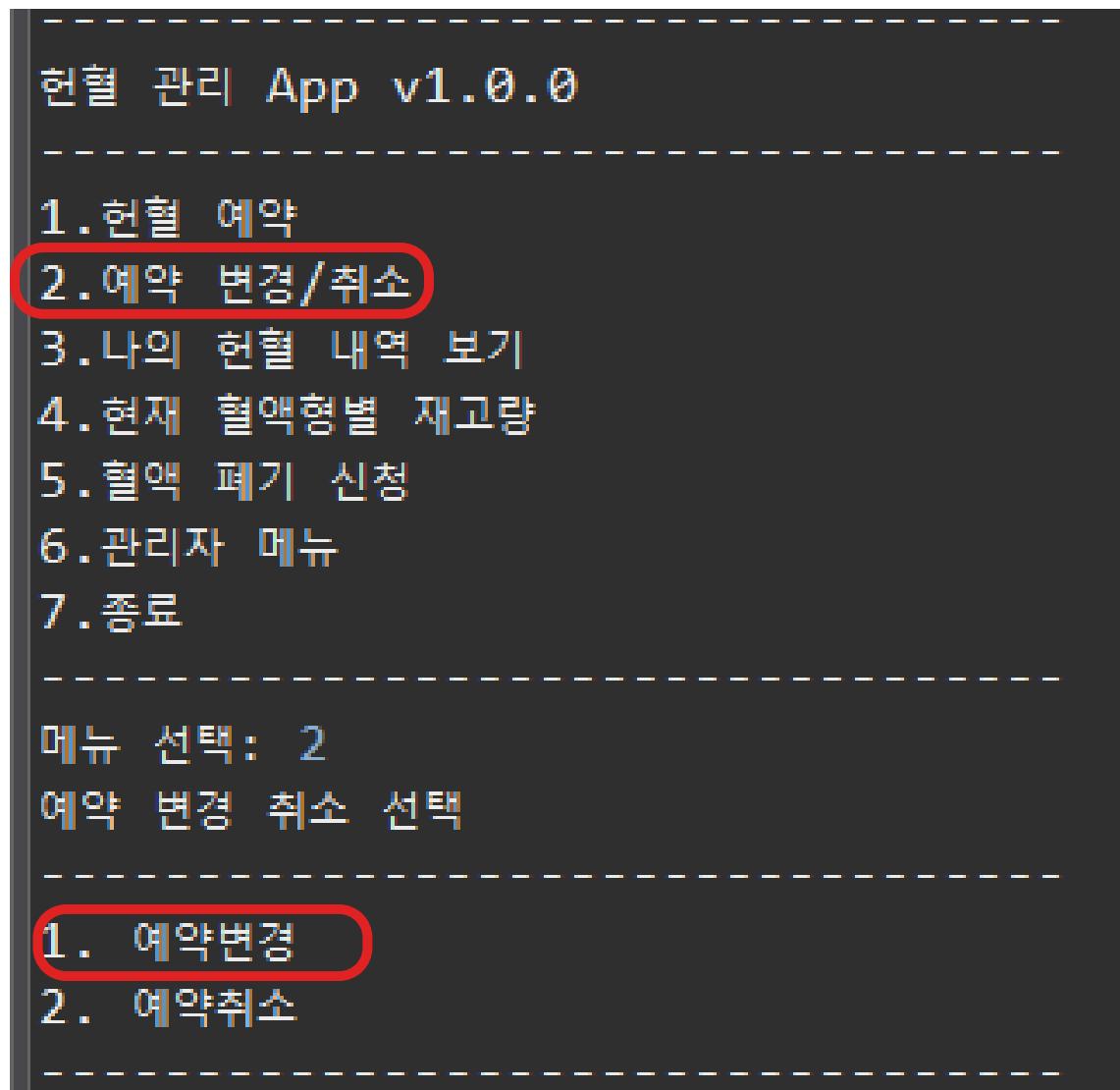
2. 수정할 현혈종류 입력

3. 변경할 예약날짜 입력

4. 변경할 사은품 입력

5. 변경된 데이터정보 확인

사용자 UI - 2. 헌혈 예약 변경



- 1.예약자 ID 입력
- 2.수정할 헌혈종류 입력
- 3.변경할 예약날짜 입력
- 4.변경할 사은품 입력
- 5.변경된 데이터정보 확인

BOOKING_ID	CENTER_ID	DONOR_ID	BO_DONATIONTYPE	BO_BLOODVOLUME	BO_BOOKTIME	BO_INTERVIEW	BO_GIFT
1 BO-1	CT01	DN-1	전혈	400	25/05/09	가능	문화상품권

사용자 UI - 2. 현혈 예약 취소

```
// 현혈 예약 취소 기능
// bookingId: 취소할 예약의 고유 ID
public void cancelDonation(String bookingId) {
    PreparedStatement pstmt = null;
    // 네 테이블 구조에 상태 컬럼이 없으므로, 해당 예약 row를 삭제하는 방식으로 처리
    // 만약 상태 컬럼을 추가한다면, UPDATE 쿼리로 변경해야 해!
    String sql = "DELETE FROM blood_booking WHERE booking_id = ?"; // <-- DELETE 쿼리로 변경하고 booking_id 컬럼 사용!

    try {
        pstmt = con.prepareStatement(sql);
        pstmt.setString(1, bookingId); // 취소할 예약 ID -> booking_id 컬럼에

        int rowsAffected = pstmt.executeUpdate(); // SQL 실행하고 삭제된 행 개수 확인

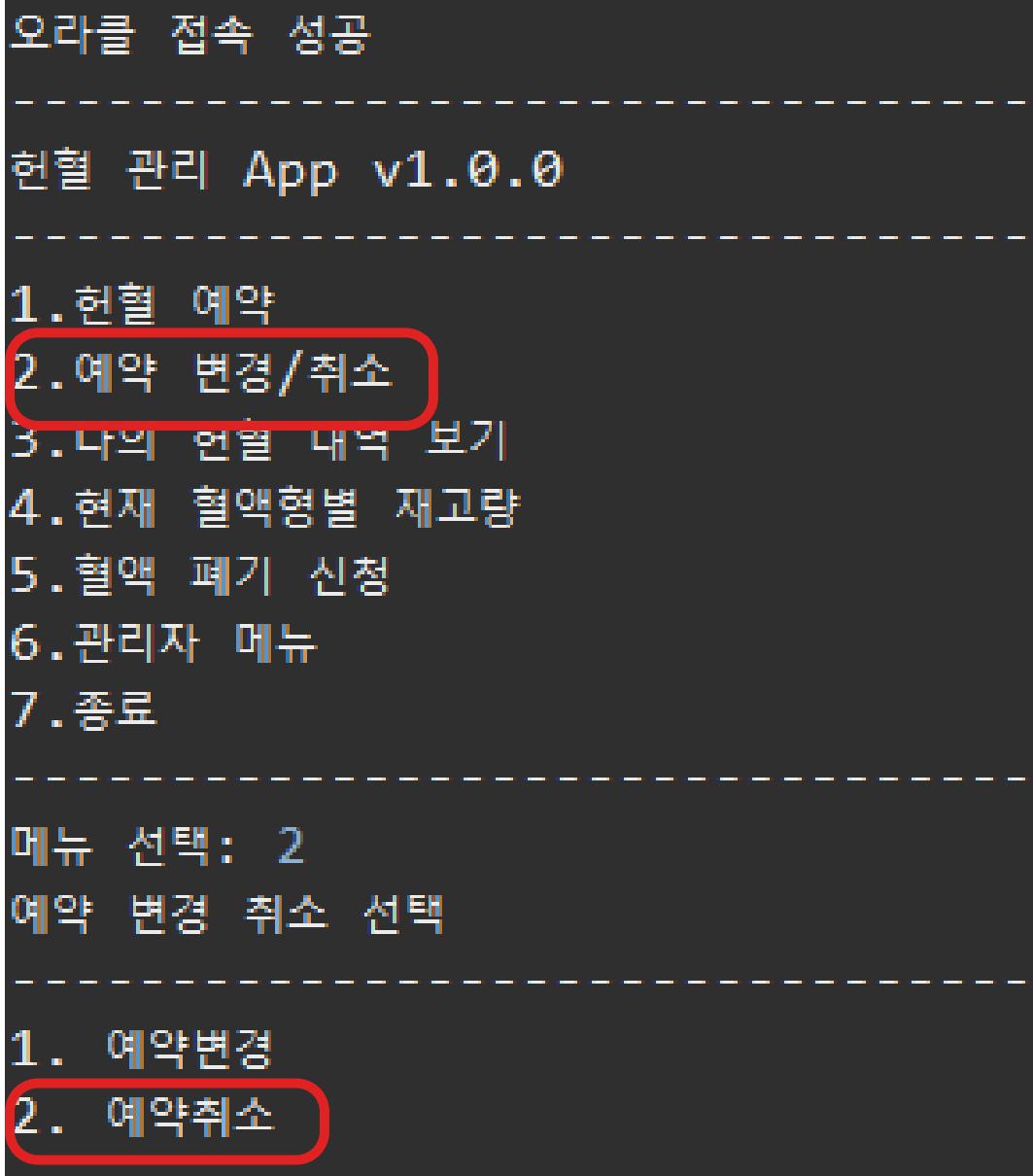
        if (rowsAffected > 0) {
            System.out.println(bookingId + " 예약이 성공적으로 취소되었습니다."); // 취소 완료!
            con.commit(); // 성공하면 저장!
        } else {
            // 해당 ID의 예약이 없을 경우
            System.out.println("예약 ID " + bookingId + "를 찾을 수 없습니다. 취소할 예약이 없어요.");
            // 이때는 뒤집힐 필요 없어.
        }
    } catch (SQLException e) {
        showErr(e); // 오류 출력
        try {
            con.rollback(); // 실패하면 되돌리기!
            System.out.println(bookingId + " 예약 취소 중 오류가 발생했어요.");
        } catch (SQLException e1) {
            e1.printStackTrace();
            System.out.println("취소 실패 후 뒤집힐 때도 오류 발생...");
        }
    } finally {
        if (pstmt != null) {
            try {
                pstmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
    // Connection은 여기서 닫으면 안 돼! DbManager에서 관리!
}
```

1. 예약자 ID 입력

- 예약된 ID가 없는 경우 메세지 출력

- 실제 현혈 불가능한 예약자의 예약이 취소된 경우 메세지 출력

사용자 UI - 2. 헌혈 예약 취소



선택: 2

--- 예약 취소---

--- 수동으로 예약한 사람중 헌혈불가능판정된경우만 예약취소 가능 ---

취소할 예약의 ID를 입력하세요(ex BO-1): BO-100

예약 ID BO-100를 찾을 수 없습니다. 취소할 예약이 없어요.

선택: 2

--- 예약 취소---

--- 수동으로 예약한 사람중 헌혈불가능판정된경우만 예약취소 가능 ---

취소할 예약의 ID를 입력하세요(ex BO-1): BO-43

BO-43 예약이 성공적으로 취소되었습니다.

1. 예약자 ID 입력
- 예약된 ID가 없는 경우 메세지 출력
 - 실제 헌혈불가능한 예약자의 예약이 취소된 경우 메세지 출력

사용자 UI - 3. 나의 헌혈 내역보기

오라클 접속 성공

헌혈 관리 App v1.0.0

1. 헌혈 예약
2. 예약 변경/취소
3. 나의 헌혈 내역 보기
4. 현재 혈액형별 재고량
5. 혈액 폐기 신청
6. 관리자 메뉴
7. 종료

메뉴 선택: 3
헌혈 아이디 입력하세요 (ex DN-1):

예약 ID: BO-3
예약 시간: 2025-05-18 00:00:00
헌혈 유형: 혈소판 성분헌혈
혈액량: 250

예약 ID: BO-8
예약 시간: 2025-05-28 00:00:00
헌혈 유형: 혈소판 성분헌혈
혈액량: 250

총 헌혈 횟수: 11회

메뉴 선택: 3
헌혈 아이디 입력하세요 (ex DN-1): DD-1
등록된 id가 존재하지 않습니다ㅠㅠ

헌혈 아이디를 입력 시

정상적인 경우에는 헌혈 기록과 함께 헌혈횟수 출력

헌혈 아이디가 없는 경우에는 실패 문구 출력

사용자 UI - 4. 현재 혈액형별 재고량

헌혈 관리 App v1.0.0

- 1. 헌혈 예약
- 2. 예약 변경/취소
- 3. 나의 헌혈 내역 보기
- 4. 현재 혈액형별 재고량
- 5. 혈액 폐기 신청

- 6. 관리자 메뉴
- 7. 종료

메뉴 선택: 4

혈액타입: O형

혈액양: 5250ml

=====

혈액타입: B형

혈액양: 6150ml

=====

혈액타입: AB형

혈액양: 2700ml

=====

혈액타입: A형

혈액양: 2300ml

```
public List<BloodJaego> bloodJaego3() {  
    List<BloodJaego> bklist = new ArrayList<>();  
  
    PreparedStatement pstmt = null;  
    String sql = "SELECT bd.do_bloodtype AS \"혈액형\", SUM(bb.bo_bloodvolume) AS \"혈액총량\" "  
        + "FROM blood_booking bb JOIN blood_donor bd ON bb.donor_id = bd.donor_id "  
        + "JOIN blood_keep bk ON bb.booking_id = bk.booking_id "  
        + "WHERE bk.status is null or bk.status = '폐기예정'" + "group by bd.do_bloodtype";  
    try {  
        pstmt = con.prepareStatement(sql);  
        rs = pstmt.executeQuery();  
  
        while (rs.next()) {  
            bklist.add(new BloodJaego(rs.getString("혈액형"), rs.getInt("혈액총량")));  
        }  
        rs.close();  
        pstmt.close();  
    } catch (SQLException e) {  
        showErr(e);  
    }  
    return bklist;  
}
```

사용자 UI 메뉴중 4번 실행
출력되는 조건(이미 사용된) 혈액을 제외한
사용 가능한 혈액(NULL), 폐기예정중인 혈액만
계산해서 재고량 출력되도록 설계

사용자 UI - 5. 혈액 폐기 신청

오라클 접속 성공

헌혈 관리 App v1.0.0

- 1. 헌혈 예약
- 2. 예약 변경/취소
- 3. 나의 헌혈 내역 보기
- 4. 현재 혈액형별 재고량
- 5. 혈액 폐기 신청**
- 6. 관리자 메뉴
- 7. 종료

5번 메뉴입력

메뉴 선택:

메뉴 선택: 5

폐기사유 항목 선택

1. 사용자요구

2. 급성 질병

3. 감염발생 지역방문

4. 기타사유

1번항목 선택

항목 선택:

항목 선택:

1

사용자요구

id를 입력해주세요(ex:B_KEY-1)

B_KEY-1

ID입력

△ 혈액 코드: B_KEY-1

㉑ 보관 시작일: 2025-04-29 00:00:00

㉑ 보관 만료일: 2025-06-03 00:00:00

☆ 폐기 사유: 사용자폐기 요구

상태: 폐기

▣ 헌혈의집 정보:

- 지점 코드: CT01

- 지점명: 동성로센터

- 주소: 대구 중구 국채보상로 598 정호빌딩 3층

- 연락처: 053-252-2285

▣ 혈액은행 정보:

- 은행 ID: BK01

- 은행명: 대구경북혈액원

- 주소: 경상북도 포항시 남구 포스코대로 404

- 연락처: 054-272-3889

헌혈 폐기 신청시 신청사유를 선택하고 ID값을 입력받아 DB에 데이터를 업데이트하고 입력한 정보값을 출력하는 과정입니다

- 여기서 상태값은 총 4가지
 - null, 폐기예정, 폐기, 사용됨
- 1) null, 폐기예정의 경우
 - 폐기, 사용됨으로 바뀔 수 있음
- 2) 상태값이 폐기이거나 사용됨 상태일 경우
 - 다른상태로 바뀌지 못함

사용자 UI - 5. 혈액 폐기 신청

```
public String bloodGetStatus(String bloodKey) { //셀렉트문으로 status값을 받아서 리턴해줌
    PreparedStatement pstmt = null;
    String sql = "select status from blood_keep where blood_key = ?";
    String status = null;

    try {
        pstmt = con.prepareStatement(sql);
        pstmt.setString(1, bloodKey);
        rs = pstmt.executeQuery();
        while (rs.next()) {
            status = rs.getString("status");
        }
        rs.close();
        pstmt.close();
    } catch (SQLException e) {
        showErr(e);
    }
    return status;
}

① public void bloodTrash(String bloodKey, String sau) { // 폐기할 고객 정보
    PreparedStatement pstmt = null;
    String status = bloodGetStatus(bloodKey);
    /////////////////////////////////////////////////
    String sql = null;
    try {
        if (status == null) {
            sql = "UPDATE blood_keep SET dis_comment = ?, status='폐기' WHERE blood_key = ?";
            pstmt = con.prepareStatement(sql);
            pstmt.setString(1, sau);
            pstmt.setString(2, bloodKey);
        } else if (status.equals("폐기예정")) {
            sql = "UPDATE blood_keep SET dis_comment = ?, status='폐기' WHERE blood_key = ?";
            pstmt = con.prepareStatement(sql);
            pstmt.setString(1, sau);
            pstmt.setString(2, bloodKey);
        }
        else {
            return;
        }
        pstmt.executeUpdate();
        pstmt.close();
        con.commit();
    } catch (SQLException e) {
        showErr(e);
    }
    try {
        con.rollback();
    } catch (SQLException e1) {
        showErr(e1);
    }
}
```

bloodkey값을 입력받아서 db에서 status(상태)값을 찾아서 리턴해줌

bloodTrash 사용전

BLOOD_KEY	END_DATE	STATUS
1 B_KEY-1	25/06/03	(null)

bloodTrash 사용후

BLOOD_KEY	END_DATE	STATUS
1 B_KEY-1	25/06/03	폐기

status값을 찾는 메서드를 활용해서
status값을 조건문을 넣어서 status값이 'null'값 이거나
'폐기예정'일 경우 '폐기'로 업데이트

이렇게 하는 이유는 status값이 수혈을 통해 '사용됨'으로
변경되었을 경우 '사용됨' 값과 중복되지 않기 위해서

사용자 UI - 5. 혈액 폐기 신청

```
public List<KeepBlood> showBloodTrash(String bloodkey) {  
    List<KeepBlood> KbList = new ArrayList<>();  
    PreparedStatement pstmt = null;  
    String sql = "select * from blood_center_bh, blood_bank_bb, blood_keep_bp where"  
        + " bp.bank_id = bb.bank_id and bp.center_id = bh.center_id and bp.blood_key= ?";  
  
    try {  
        pstmt = con.prepareStatement(sql);  
        pstmt.setString(1, bloodkey);  
        rs = pstmt.executeQuery();  
        while (rs.next()) {  
            BloodCenter bh = new BloodCenter(rs.getString("center_id"), rs.getString("c_name"),  
                rs.getString("c_address"), rs.getString("c_tel"));  
  
            BloodBank bb = new BloodBank(rs.getString("bank_id"), rs.getString("bk_name"), rs.getString("bk_addr"),  
                rs.getString("bk_tel"));  
  
            KeepBlood kb = new KeepBlood(rs.getString("blood_key"), bh, bb, rs.getString("st_date"),  
                rs.getString("end_date"), rs.getString("status"), rs.getString("dis_comment"));  
  
            KbList.add(kb);  
        }  
        rs.close();  
        pstmt.close();  
  
        con.commit();  
    } catch (SQLException e) {  
        showErr(e);  
        try {  
            con.rollback();  
        } catch (SQLException e1) {  
            showErr(e1);  
        }  
    }  
    return KbList;  
}
```

bloodkey값을 입력받아 db에서 테이블을 조인하고
데이터를 불러와서 커서를 통해 데이터값을
각각의 객체에 집어넣고
최종적으로 KbList 값에 넣어서 리턴하는 과정입니다.

사용자 UI - 5. 혈액 폐기 신청

```
case 5:  
    while (true) {  
        int n = Menu.DeleteMenu(s);  
  
        if (n == 1) {  
            System.out.println("사용자요구");  
            System.out.println("id를 입력해주세요(ex:B_KEY-1)");  
            String bloodkey = s.next();  
            String sau = "사용자폐기 요구";  
            update ① db.bloodTrash(bloodkey, sau); ② list.add  
            List<KeepBlood> KbList = db.showBloodTrash(bloodkey);  
            for (KeepBlood kb : KbList) {  
                show ③ kb.showBloodAll();  
            }  
        }  
        public void showBloodAll() {  
            System.out.println("① 혈액 코드: " + bldKey);  
            System.out.println("② 보관 시작일: " + keepStDate);  
            System.out.println("③ 보관 만료일: " + keepEndDate);  
            System.out.println("④ 폐기 사유: " + comment);  
            System.out.println("상태: " + status);  
  
            if (bh != null) {  
                System.out.println("⑤ 헌혈의집 정보:");  
                System.out.println(" - 지점 코드: " + bh.getcId());  
                System.out.println(" - 지점명: " + bh.getcName());  
                System.out.println(" - 주소: " + bh.getcAddr());  
                System.out.println(" - 연락처: " + bh.getcTel());  
            }  
  
            if (bk != null) {  
                System.out.println("⑥ 혈액은행 정보:");  
                System.out.println(" - 은행 ID: " + bk.getBkId());  
                System.out.println(" - 은행명: " + bk.getBkname());  
                System.out.println(" - 주소: " + bk.getBkAddr());  
                System.out.println(" - 연락처: " + bk.getBkTel());  
            }  
        }  
    }  
}
```

결과



항목 선택:

1

사용자요구

id를 입력해주세요(ex:B_KEY-1)

B_KEY-1

① 혈액 코드: B_KEY-1

② 보관 시작일: 2025-04-29 00:00:00

③ 보관 만료일: 2025-06-03 00:00:00

④ 폐기 사유: 사용자폐기 요구

상태: 폐기

⑤ 헌혈의집 정보:

- 지점 코드: CT01

- 지점명: 동성로센터

- 주소: 대구 중구 국채보상로 598 정호빌딩 3층

- 연락처: 053-252-2285

⑥ 혈액은행 정보:

- 은행 ID: BK01

- 은행명: 대구경북혈액원

- 주소: 경상북도 포항시 남구 포스코대로 404

- 연락처: 054-272-3889

메서드를 통해 리턴받은 ArrayList값을
for each 반복문을 통해
클래스에 미리 만들어둔 showBloodAll()
메소드를 사용하여 출력해줍니다

관리자 UI

오라클 접속 성공

현혈 관리 App v1.0.0

- 1. 현혈 예약
- 2. 예약 변경/취소
- 3. 나의 현혈 내역 보기
- 4. 현재 혈액형별 재고량
- 5. 혈액 폐기 신청
- 6. 관리자 메뉴
- 7. 종료

메뉴 선택: 6

메뉴 선택: 6

id를 입력해주세요 admin

pw를 입력해주세요 1234

관리자 ID 및 비밀번호를 입력해야만 하위 메뉴 동작

```
public void subMenu(Scanner d) throws Exception {  
    System.out.print("id를 입력해주세요 ");  
    String id = d.next();  
    System.out.print("pw를 입력해주세요 ");  
    String pw = d.next().toString();  
  
    boolean aa = true;  
    int n=0;  
  
    if(id.equals("admin") && pw.equals("1234")) {  
        while(aa) {  
            System.out.println("====관리자 메뉴====");  
            System.out.println("1. 사용자 데이터 추가");  
            System.out.println("2. 사용자 예약 데이터 추가");  
            System.out.println("3. 혈액형별 현혈자 검색");  
            System.out.println("4. 현혈예약 정보 전체검색");  
            System.out.println("5. 기간이 만료된 혈액 폐기");  
            System.out.println("6. 수동으로 수혈 날짜입력");  
            System.out.println("7. 이전 메뉴로 돌아가기");  
            System.out.println("-----");  
            System.out.print("항목 선택: ");  
            int m = d.nextInt();  
  
            switch(m) {  
                case 1:  
                case 2:  
                case 3:  
                case 4:  
                case 5:  
                case 6:  
                case 7:  
                    aa = false;  
                    break;  
                default:  
                    System.out.println("잘못된 항목입니다.");  
            }  
        }  
    }  
}
```

관리자 UI - 1. 사용자 데이터 추가

====관리자 메뉴====

1. 사용자 데이터 추가

2. 사용자 예약 데이터 추가

3. 혈액형별 헌혈자 검색

4. 헌혈예약 정보 전체검색

5. 기간이 만료된 혈액 폐기

6. 수동으로 수혈 날짜입력

7. 사용 가능 혈액 상세보기

8. 이전 메뉴로 돌아가기

생성할 사람의 수를 입력하세요 200

◆ DONOR_ID	DO_NAME	DO_RRNUMBER	DO_GENDER	DO_ADDRESS	DO_TEL	DO_BLOODTYPE
DN-199	이재율	680123-2477120	여	대구광역시 달성군 하빈면	010-2537-3559	O형
DN-200	강진진	600201-2273022	여	대구광역시 수성구 파동	010-9954-9537	O형
DN-201	양은홍	091005-4853424	여	대구광역시 달성군 유가읍	010-9322-0354	O형

◆ DONEE_ID	DEE_NAME	DEE_RRNUMBER	DEE_GENDER	DEE_ADDRESS	DEE_TEL	DEE_BLOODTYPE
DE-197	양유록	050501-3139626	남	대구광역시 달성군 다사읍	010-5582-2448	AB형
DE-198	박진진	731118-1040725	남	대구광역시 수성구 파동	010-3557-6846	O형
DE-199	박서우	750408-1674532	남	대구광역시 북구 읍내동	010-2978-9735	B형
DE-200	강민수	060227-4117635	여	대구광역시 달서구 송현동	010-9592-0196	A형
DE-201	오수경	030803-4105923	여	대구광역시 군위군 부계면	010-9928-5389	O형

```
System.out.print("생성할 사람의 수를 입력하세요");
n=d.nextInt();

//DONOR 생성
String[] rrlist = new String[n];
String[] randtel = new String[n];
rrlist= r.getRandRr(n);          //랜덤 주민등록번호 n개 생성
randtel = r.getRandTel(n);        //랜덤 전화번호 n개 생성
for(int i=0;i<n;i++) {
    Donor dn = new Donor(db.getdid(),r.getRandomName2() ,
    rrlist[i],r.getGender2(rrlist[i]) , r.getAddress2(),
    randtel[i], r.getBloodType2());
    db.insertDonor(dn);           //헌혈자 정보 데이터베이스에 인서트
}
```

생성할 사람 수를 입력 →
헌혈자와 수혈자 랜덤 데이터 정보가
입력 개수만큼 만들어짐

<특징>

- 주민등록번호와 전화번호는 중복 불가능함
- 주민등록번호와 성별을 매치시킴

관리자 UI - 2. 사용자 예약 데이터 추가

== 관리자 메뉴 ==	
1. 사용자 데이터 추가	
2. 사용자 예약 데이터 추가	
3. 혈액형별 헌혈자 검색	
4. 헌혈예약 정보 전체검색	
5. 기간이 만료된 혈액 폐기	
6. 수동으로 수혈 날짜입력	
7. 사용 가능 혈액 상세보기	
8. 이전 메뉴로 돌아가기	

헌혈예약

BOOKING_ID	CENTER_ID	DONOR_ID	BO_DONATIONTYPE	BO_BLOODVOLUME	BO_BOOKTIME	BO_INTERVIEW	BO_GIFT
149 BO-83	CT07	DN-67	혈소판 성분헌혈	250	25/04/06	가능	기념품을 선택하지 않겠습니다.
150 BO-84	CT01	DN-9	혈장 성분헌혈	500	25/04/12	가능	기념품을 선택하지 않겠습니다.
151 BO-85	CT09	DN-81	혈소판 성분헌혈	250	25/05/27	가능	기념품을 선택하지 않겠습니다.

혈액정보

BLOOD_KEY	BOOKING_ID	CENTER_ID	BANK_ID	ST_DATE	END_DATE	DIS_COMMENT	STATUS	DE_BOOKING_ID
B_KEY-149	BO-149	CT05	BK01	25/03/21	25/04/25	(null)	(null)	(null)
B_KEY-150	BO-150	CT06	BK01	25/05/26	25/06/30	(null)	(null)	(null)
B_KEY-151	BO-151	CT03	BK01	25/04/13	25/05/18	(null)	(null)	(null)

수혈

DE_BOOKING_ID	DONEE_ID	BANK_ID	DE_DONATIONTYPE	DE_BLOODVOLUME	DE_BOOKTIME
3 DE_BO-3	DE-193	BK01	혈소판 성분헌혈	250	25/04/19
4 DE_BO-4	DE-60	BK01	혈장 성분헌혈	1000	25/04/30
5 DE_BO-5	DE-73	BK01	혈장 성분헌혈	1000	25/06/04

항목 선택: 2

생성할 헌혈 예약 수를 입력하세요 150

생성할 수혈 예약 수를 입력하세요 3

생성할 헌혈 예약 수를 입력 ->

헌혈 예약을 랜덤생성. 동시에 혈액정보에 저장.

생성할 수혈 예약 수를 입력->

수혈 랜덤 생성.

동시에 혈액정보의 혈액을 사용됨 표기. (관련 내용 뒷장)

관리자 UI - 2. 사용자 예약 데이터 추가

```
case 2 :  
    System.out.print("생성할 헌혈 예약 수를 입력하세요");  
    int n2=d.nextInt();  
    System.out.print("생성할 수혈 예약 수를 입력하세요");  
    int bkn=d.nextInt();  
    for(int i=0;i<n2;i++) {  
        String bookid = db.getNextBookingId(); //부킹id (헌혈예약과 혈액정보 둘다 필요한 값. 랜덤생성한 정보를 고정.)  
        String centerid = r.getRandomCenterId(); //센터id (위와 같음)  
        String booktime = r.getRandomBookTime(); //예약시간(위와 같음)  
        String dtype = r.getRandomDonationType();  
        //book: 부킹id, 센터id, donorid, donationtype, bloodvolume, booktime, interview, gift  
        1. db.RandBookingFull(bookid,centerid, r.getRandomDonorId(n), dtype, r.getBloodVolume(dtype), booktime,  
            "가능", r.getRandomGift());  
        //keep: bloodkey, 부킹id, 센터id, bankid, stdate, enddate, comment  
        2. db.addBloodKeep(bookid, centerid, "BK01", booktime, null);  
    }  
  
    //DONEE_BOOKING  
    int randnum=0;  
    for(int i=0;i<bkn;i++) {  
        String dtype = r.getRandomDonationType(); //헌혈종류 (혈액량 구할 때 사용하기 위해 랜덤생성한 정보 고정.)  
        String deeId = r.getRandomDoneeId(n); //DONEE ID(수혈받을 사용자 랜덤생성)  
        randnum = ((int)(Math.random()*2)+1); //필요한 수량 (1~2개 랜덤)  
  
        3. DoneeBooking bk = new DoneeBooking(db.getDoneeBookingId(),  
            deeId, "BK01", dtype,  
            r.getBloodVolume(dtype)*randnum, r.getRandomBookTime24donee());  
            //필요한 혈액량을 랜덤하게 지정 //수혈받을 날짜 랜덤지정  
  
        4. db.insertDoneeBooking(bk); //수혈정보 데이터베이스에 인서트  
  
        5. ArrayList<String> test1 = db.getDoneeBloodSK(deeId); //수혈자가 사용할 수 있는 혈액재고 찾기  
  
        6. for(int j=0;j<randnum;j++) {  
            db.updateKeepDonee(deeId, test1.get(j)); //사용한 혈액에 사용날짜와 사용됨 정보 업데이트  
        }  
    }
```

생성할 헌혈 예약 수를 입력 ->
헌혈 예약을 랜덤생성. 동시에 혈액정보에 저장.

전혈 : 400ml
혈소판 성분헌혈 : 250ml
혈장 성분헌혈 : 500ml

헌혈예약 생성 및 데이터베이스 인서트

예약일자에 헌혈함을 가정
헌혈 받은 혈액에 관한

혈액정보 생성 및 데이터베이스 인서트

생성할 수혈 예약 수를 입력->
수혈 랜덤 생성.
동시에 혈액정보의 혈액을 사용됨 표기.

수혈정보 랜덤 생성

수혈정보 인서트

사용가능 혈액재고 찾기
사용한 혈액에 정보 업데이트
(관련내용 뒷장)

관리자 UI - 2. 사용자 예약 데이터 추가

```
ArrayList<String> test1 = db.getDoneeBloodSK(deeId); //수혈자가 사용할 수 있는 혈액재고 찾기
```

사용가능 혈액재고 찾기

```
public ArrayList<String> getDoneeBloodSK(String DoneeId) {  
    PreparedStatement pstmt1 = null;  
    PreparedStatement pstmt2 = null;  
    PreparedStatement pstmt3 = null;  
  
    String bloodtype = null;  
    Date deBooktime = null;  
    String deDonationType = null;  
    String keepBloodId = null;  
    ArrayList<String> keepId = new ArrayList<String>();
```

sql 1: 수혈자의 혈액형 찾기

```
String sql1 = "select DEE_BLOODTYPE from blood_donee where DONEE_ID = ? ";  
String sql2 = "select DE_BOOKTIME,DE_DONATIONTYPE from donee_booking where DONEE_ID = ? ";  
String sql3 = "select 혈액관리번호 from blood_all where 혈액형 = ? \r\n"  
+ " and to_date(보관일자,'YY/MM/DD')<to_date(?, 'YY/MM/DD') \r\n"  
+ " and to_date(?, 'YY/MM/DD')<=to_date(만료일자, 'YY/MM/DD') \r\n"  
+ " and 헌혈종류 = ? order by 만료일자 asc ";  
  
try {  
    pstmt1 = con.prepareStatement(sql1);  
    pstmt1.setString(1, DoneeId); //혈액형 찾을 수혈자코드  
    rs = pstmt1.executeQuery();  
    if(rs.next()) {  
        bloodtype = rs.getString("DEE_BLOODTYPE");  
    }  
    rs.close();  
    pstmt1.close();
```

sql 2: 수혈자의 수혈 시간과
필용한 혈액종류 찾기

```
    pstmt2 = con.prepareStatement(sql2);  
    pstmt2.setString(1, DoneeId); //예약시간, 수혈종류 찾을 수혈자코드  
    rs = pstmt2.executeQuery();  
    if(rs.next()) {  
        deBooktime = rs.getDate("DE_BOOKTIME");  
        deDonationType = rs.getString("DE_DONATIONTYPE");  
    }  
    rs.close();  
    pstmt2.close();
```

sql 1: bloodtype에 수혈자 혈액형 정보 저장

```
pstmt3 = con.prepareStatement(sql3);  
pstmt3.setString(1, bloodtype);  
pstmt3.setDate(2, deBooktime);  
pstmt3.setDate(3, deBooktime);  
pstmt3.setString(4, deDonationType);  
rs = pstmt3.executeQuery();
```

return keepId;

sql 3: sql1,2를 통해 얻은 정보를 통해서
혈액형과 혈액종류가 일치하며
수혈날짜에 사용가능한 혈액을 찾아서
혈액재고의 혈액관리번호들을 ArrayList에 저장하여 리턴

관리자 UI - 2. 사용자 예약 데이터 추가

--가상테이블 만들기

```
create or replace view blood_all as
    select bk.blood_key "혈액관리번호", bk.booking_id "예약번호", bk.center_id "지점코드",
    bk.bank_id "은행코드",
    bd.donor_id "헌혈자코드", bd.do_bloodtype "혈액형", bb.bo_donationtype "헌혈종류",
    bb.bo_bloodvolume "헌혈량",
    bk.st_date "보관일자", bk.end_date "만료일자", bk.status "상태", bk.dis_comment "폐기사유"
    from blood_donor bd, blood_booking bb, blood_keep bk
    where bk.booking_id = bb.booking_id and bb.donor_id = bd.donor_id;
//예시
select * from blood_all;
```

결과 x

SQL | 50개의 행이 인출됨(0,055초)

혈액관리번호	예약번호	지점코드	은행코드	헌혈자코드	혈액형	헌혈종류	헌혈량	보관일자	만료일자	상태	폐기사유
B_KEY-1	BO-1	CT01	BK01	DN-1	O형	전혈	400	25/04/29	25/06/03	(null)	(null)
B_KEY-2	BO-2	CT04	BK01	DN-60	A형	전혈	400	25/05/21	25/06/25	(null)	(null)

```
String sql1 = "select DEE_BLOODTYPE from blood_donee where DONEE_ID = ? ";
String sql2 = "select DE_BOOKTIME,DE_DONATIONTYPE from donee_booking where DONEE_ID = ? ";
String sql3 = "select 혈액관리번호 from blood_all where 혈액형 = ? \r\n"
    + "    and to_date(보관일자,'YY/MM/DD')<to_date(?,'YY/MM/DD') \r\n"
    + "    and to_date(?,'YY/MM/DD')<=to_date(만료일자,'YY/MM/DD') \r\n"
    + "    and 헌혈종류 = ? order by 만료일자 asc ";
```

헌혈자, 헌혈예약, 혈액정보를 포함하는
가상테이블을 만들어서
필요한 정보 검색에 사용

관리자 UI - 2. 사용자 예약 데이터 추가

```
for(int j=0;j<randnum;j++) {  
    db.updateKeepDonee(deeId, test1.get(j));  
} //사용한 혈액에 사용날짜와 사용됨 정보 업데이트
```

```
public void updateKeepDonee(String DoneeId, String Blood_key) {  
    PreparedStatement pstmt = null;  
  
    String sql = "UPDATE blood_keep\r\n"  
    + "SET status = '사용됨',\r\n"  
    + "    dis_comment = (SELECT de_booktime \r\n"  
    + "                    FROM donee_booking \r\n"  
    + "                   WHERE donee_id = ? AND ROWNUM = 1),\r\n"  
    + "    de_booking_id = (SELECT de_booking_id \r\n"  
    + "                      FROM donee_booking \r\n"  
    + "                     WHERE donee_id = ? AND ROWNUM = 1)\r\n"  
    + "WHERE blood_key = ?";  
  
    try {  
        pstmt = con.prepareStatement(sql);  
        pstmt.setString(1, DoneeId); //수혈번호  
        pstmt.setString(2, DoneeId); //수혈자코드  
        pstmt.setString(3, Blood_key);  
        pstmt.executeUpdate();  
        con.commit();  
        System.out.println("데이터 입력 성공!");  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

test1: 사용가능한 혈액의 혈액관리번호들
필요한 개수만큼 test1에 있는 혈액정보를
사용됨으로 업데이트
(만료날짜 촉박한 것부터 사용)

상태에는 “사용됨” 업데이트
코멘트에 사용날짜(=수혈한 날짜)를 업데이트
수혈번호 입력(어떤 수혈에 사용되었는지)
사용할 혈액관리번호

관리자 UI - 2. 사용자 예약 데이터 추가

==관리자 메뉴==							
1.	사용자 데이터 추가						
2.	사용자 예약 데이터 추가						
3.	혈액형별 헌혈자 검색						
4.	헌혈예약 정보 전체검색						
5.	기간이 만료된 혈액 폐기						
6.	수동으로 수혈 날짜입력						
7.	사용 가능 혈액 상세보기						
8.	이전 메뉴로 돌아가기						

수혈

DE_BOOKING_ID	DONEE_ID	BANK_ID	DE_DONATIONTYPE	DE_BLOODVOLUME	DE_BOOKTIME
3 DE_BO-3	DE-193	BK01	혈소판 성분헌혈	250	25/04/19
4 DE_BO-4	DE-60	BK01	혈장 성분헌혈	1000	25/04/30
5 DE_BO-5	DE-73	BK01	혈장 성분헌혈	1000	25/06/04

수혈에서 필요한 혈액량을 참고해 혈액정보에 반영

전혈 : 400ml

혈소판 성분헌혈 : 250ml

혈장 성분헌혈 : 500ml

예) 혈장 성분헌혈 1팩 500ml이며, 1000ml 필요 시 혈액정보에 2팩을 사용됨 업데이트

<고려사항>

1. 수혈자의 혈액형과 일치하는 혈액
2. 필요한 종류의 혈액 및 양
3. 수혈하는 날짜에 사용가능한 혈액 (그중에서 보관기간이 얼마 안남은 것부터 사용)

혈액정보

BLOOD_KEY	BOOKING_ID	CENTER_ID	BANK_ID	ST_DATE	END_DATE	DIS_COMMENT	STATUS	DE_BOOKING_ID
1 B_KEY-111	BO-111	CT05	BK01	25/03/25	25/04/29	25/04/19	사용됨	DE_BO-3
2 B_KEY-18	BO-18	CT04	BK01	25/04/29	25/06/03	25/04/30	사용됨	DE_BO-4
3 B_KEY-99	BO-99	CT09	BK01	25/04/08	25/05/13	25/04/30	사용됨	DE_BO-4
4 B_KEY-75	BO-75	CT01	BK01	25/05/09	25/06/13	25/06/04	사용됨	DE_BO-5
5 B_KEY-143	BO-143	CT06	BK01	25/05/01	25/06/05	25/06/04	사용됨	DE_BO-5

관리자 UI - 3. 혈액형 별 헌혈 내역

```
public void getDonorsByBloodType(String bloodType) {  
    PreparedStatement pstmt = null;  
    String sql = "SELECT * FROM blood_donor WHERE do_bloodtype = ?";  
  
    try {  
        pstmt = con.prepareStatement(sql);  
  
        pstmt.setString(1, bloodType);  
  
        rs = pstmt.executeQuery();  
  
        boolean hasData = false;  
  
        System.out.println("== " + bloodType + " 혈액형 헌혈자 목록 ==");  
  
        while (rs.next()) {  
            hasData = true;  
            System.out.println("헌혈자 ID: " + rs.getString("donor_id"));  
            System.out.println("이름: " + rs.getString("do_name"));  
            System.out.println("주민등록번호: " + rs.getString("do_rrnumber"));  
            System.out.println("성별: " + rs.getString("do_gender"));  
            System.out.println("주소: " + rs.getString("do_address"));  
            System.out.println("전화번호: " + rs.getString("do_tel"));  
            System.out.println("혈액형: " + rs.getString("do_bloodtype"));  
            System.out.println("-----");  
        }  
  
        if (!hasData) {  
            System.out.println("해당 혈액형의 헌혈자가 없습니다.");  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        if (pstmt != null) {  
            try {  
                pstmt.close();  
            } catch (SQLException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

관리자 메뉴가
필요한 상황

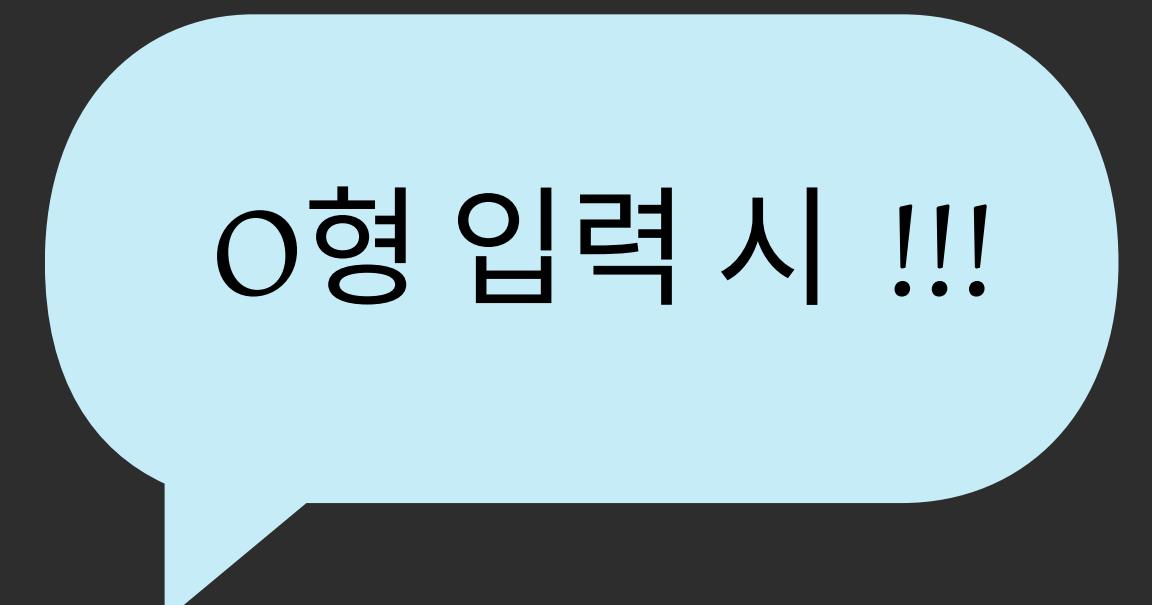
1. 응급 수혈이 필요한 상태
2. 희귀 혈액형 보유자의 수술 또는 치료
3. 산모와 태아 간 혈액형 부적합

관리자 UI - 3. 혈액형 별 헌혈 내역

```
143  
144     case 3 : {  
145         System.out.print("검색할 혈액형을 입력해주세요(A형, B형, O형, AB형): ");  
146         String bloodtype = d.next();  
147         db.getDonorsByBloodType(bloodtype);  
148     }  
149     break;  
150 }  
151 case 4 : {  
152     db.showbooking();  
153 }
```

Console ×
BloodMain [Java Application] C:\eclipse-jee-2025-03-R-win32-x86_64\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe (2025. 5. 9. 오후 2:36:05 elapsed: 0:01:12) [p]
3. 혈액형별 헌혈자 검색
4. 헌혈예약 정보 전체검색
5. 기간이 만료된 혈액 폐기
6. 수동으로 수혈 날짜입력
7. 이전 메뉴로 돌아가기

항목 선택: 3
검색할 혈액형을 입력해주세요(A형, B형, O형, AB형): O형
==== O형 혈액형 헌혈자 목록 ====
헌혈자 ID: DN-1
이름: 홍길동
주민등록번호: 990101-1234567
성별: 남
주소: 대구 신암1동
전화번호: 010-1111-1111
혈액형: O형



O형 입력 시 !!!

관리자 UI – 6. 기간이 만료된 혈액 폐기

```
public void adminBloodTrash() {  
    PreparedStatement pstmt1 = null;  
    PreparedStatement pstmt2 = null;  
  
    String sql1 = "UPDATE blood_keep SET status = '폐기' where (status is null or status ='폐기예정')\r\n" + " and to_date(end_date,'YY/MM/DD')<to_date(sysdate,'YY/MM/DD')";  
    String sql2 = "UPDATE blood_keep SET status = '폐기예정' WHERE status is null and\r\n" + " to_date(end_date,'YY/MM/DD')>=to_date(sysdate,'YY/MM/DD') and\r\n" + " to_date(end_date,'YY/MM/DD')<= to_date(sysdate,'YY/MM/DD')+7";  
  
    try {  
        pstmt1 = con.prepareStatement(sql1);  
        int count1 = pstmt1.executeUpdate();  
        pstmt1.close();  
  
        pstmt2 = con.prepareStatement(sql2);  
        int count2 = pstmt2.executeUpdate();  
        pstmt2.close();  
  
        con.commit();  
  
        System.out.println("✓ 폐기 처리된 혈액 수: " + count1);  
        System.out.println("⚠ 폐기 예정된 혈액 수: " + count2);  
    } catch (SQLException e) {  
        showErr(e);  
        try {  
            con.rollback();  
        } catch (SQLException e1) {  
            showErr(e1);  
        }  
    }  
}
```

사용되지 않거나 폐기예정인 혈액 대상

사용되지 않은 혈액 대상

sql 문이 한번 실행될 때마다
변수값에 숫자로 넣어준 뒤 출력

sql1) 폐기

오늘날짜(sysdate)가 만료일(end_date)을
지났을 경우 '폐기'

sql2) 폐기예정

A: 오늘날짜(sysdate)

B: 만료일(end_date)

A <= B <= A+7

EX) 오늘날짜가 5.12일 경우

5.12 <= B <= 5.19

만료일이 오늘날짜에서 오늘날짜 +7일인 날짜
사이의 값을 '폐기예정'으로 바꿈

관리자 UI - 6. 기간이 만료된 혈액 폐기

혈액 폐기 메뉴 사용전

END_DATE	DIS_COMMENT	BOOKING_ID	STATUS
25/04/30	(null)	BO-16	(null)
25/05/02	(null)	BO-30	(null)
25/05/02	(null)	BO-24	(null)
25/05/03	(null)	BO-3	(null)
25/05/05	(null)	BO-9	(null)
25/05/06	(null)	BO-21	(null)
25/05/06	(null)	BO-7	(null)
25/05/10	(null)	BO-29	(null)
25/05/10	(null)	BO-40	(null)
25/05/11	(null)	BO-8	(null)
25/05/16	(null)	BO-22	(null)
25/05/19	(null)	BO-51	(null)
25/05/20	(null)	BO-42	(null)
25/05/20	(null)	BO-5	(null)
25/05/23	(null)	BO-43	(null)
25/05/24	(null)	BO-32	(null)
25/05/28	(null)	BO-48	(null)
25/05/28	(null)	BO-18	(null)
25/05/29	(null)	BO-38	(null)
25/05/29	(null)	BO-23	(null)
25/06/01	(null)	BO-19	(null)

====관리자 메뉴=====

1. 사용자 데이터 추가
2. 사용자 예약 데이터 추가
3. 혈액형별 헌혈자 검색
4. 헌혈예약 정보 전체검색
5. 기간이 만료된 혈액 폐기
6. 수동으로 수혈 날짜입력
7. 사용 가능 혈액 상세보기
8. 이전 메뉴로 돌아가기

항목 선택: 5

폐기 처리된 혈액 수: 10
 폐기 예정된 혈액 수: 2

폐기처리되었습니다

혈액 폐기 메뉴 사용 후

(sysdate) 05/12일에 실행

END_DATE	DIS_COMMENT	BOOKING_ID	STATUS
25/04/30	(null)	BO-16	폐기
25/05/02	(null)	BO-30	폐기
25/05/02	(null)	BO-24	폐기
25/05/03	(null)	BO-3	폐기
25/05/05	(null)	BO-9	폐기
25/05/06	(null)	BO-21	폐기
25/05/06	(null)	BO-7	폐기
25/05/10	(null)	BO-29	폐기
25/05/10	(null)	BO-40	폐기
25/05/11	(null)	BO-8	폐기
25/05/16	(null)	BO-22	폐기예정
25/05/19	(null)	BO-51	폐기예정
25/05/20	(null)	BO-42	(null)
25/05/20	(null)	BO-5	(null)
25/05/23	(null)	BO-43	(null)
25/05/24	(null)	BO-32	(null)
25/05/28	(null)	BO-48	(null)
25/05/28	(null)	BO-18	(null)
25/05/29	(null)	BO-38	(null)
25/05/29	(null)	BO-23	(null)
25/06/01	(null)	BO-19	(null)

5.12~5.19사이의 값은 폐기예정처럼됨

관리자 UI - 7. 사용 가능 혈액 상세보기

====관리자 메뉴====

1. 사용자 데이터 추가
2. 사용자 예약 데이터 추가
3. 혈액형별 헌혈자 검색
4. 헌혈예약 정보 전체검색
5. 기간이 만료된 혈액 폐기
6. 수동으로 수혈 날짜입력
7. 사용 가능 혈액 상세보기
8. 이전 메뉴로 돌아가기

1. 시스템 날짜 기준으로 오늘 사용가능한 혈액만 표시
2. 혈액형과 혈액종류별로 보이도록 표시

항목 선택: 7

혈액형: AB형

혈액종류: 전혈

혈액량: 2400ml

혈액형: AB형

혈액종류: 혈소판 성분헌혈

혈액량: 250ml

혈액형: AB형

혈액종류: 혈장 성분헌혈

혈액량: 2000ml

혈액형: A형

혈액종류: 전혈

혈액량: 1200ml

혈액형: A형

혈액종류: 혈소판 성분헌혈

혈액량: 1000ml

혈액형: A형

혈액종류: 혈장 성분헌혈

혈액량: 3500ml

조건: 만료 일자가 오늘(sysdate)과 오늘(sysdate)+35일 사이인 혈액 중
사용 가능한 상태인 혈액
(* 보관 기간: 헌혈일부터 +35일로 설정)

자체 평가

손한이

상속화, 다형성 등에 대한 개념을 배웠지만
실질적으로 프로젝트에 사용된 요소는 많지
않았다고 생각합니다. 기획에서 생각했던 바와
실제 설계 과정에서 생성자와 변수간의 연결 등을
어떻게 적용할 수 있을지 더 많은 연구와
시행착오가 필요하다고
생각하게 되는 시간이었습니다.

윤정서

계속해서 프로젝트를 경험하면서 처음 정했던 구조대로 가는 경우는 거의 없겠지만 처음에 이상한 구조를 가지고 작업을 계속 해서 하다 보면 다양한 오류가 생길 수 있음을 알게 되었습니다.

전현식

웹개발을 공부를 시작하고 여러 팀 프로젝트를 진행하면서 이번에 처음으로 팀장을 도전하여서 걱정이 많았지만 다들 적극적으로 참여해주셔서 프로젝트를 진행하는데 걱정했던거보다 원활하게 진행할수있었습니다.

양수경

충분한 데이터를 위해서 모든 랜덤 데이터를 담당했더니, 매소드 만들기에 익숙해진 것 같습니다.
다만 주로 메소드를 만들고 변수를 받아 사용해서 객체지향설계를 못한 것 같아 아쉬웠습니다.

피동기

메서드를 만드는데 계속 오류가 생겨서 오류를 찾아내고 수정하는데 많은 시간이 걸렸던 것 같습니다,
같은 작업을 계속 반복하고 디버깅을 하면서 오류를 찾는데 인내심을 많이 기를 수 있었던 것 같습니다

프로젝트 사연



Q & A



THANK YOU

