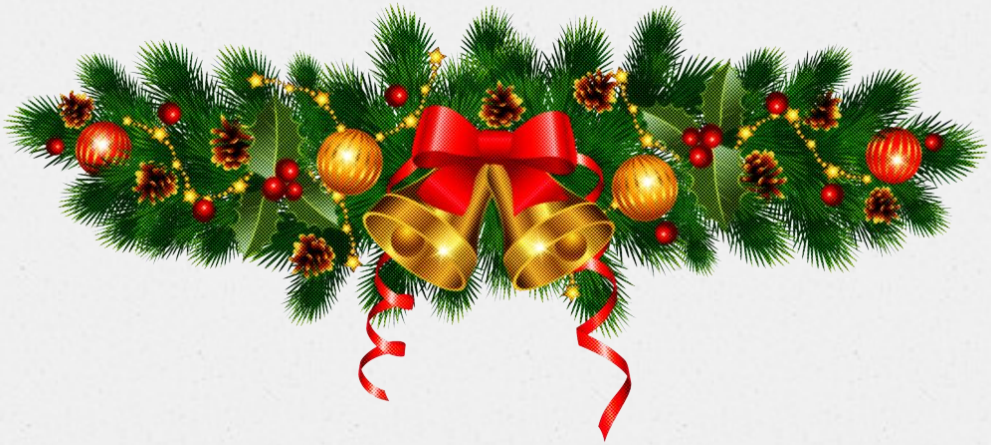
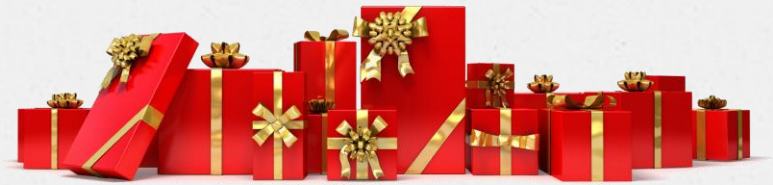


도서관2팀 백정하



프로젝트 수행 일정

- 위키에 따르면 총 11일을 진행했습니다.

- Day 1-2 : gitlab에 리포지토리 생성
- Day 2-9 : 본격적인 프로젝트
- Day 10 : 배포
- Day 11 : 발표 준비

한 것 / 안한 것

- 프로젝트에 제시된 필수 기능들
- 선택 기능 중
- 아이디는 이메일 형식만 받기
- 현재 재고가 없을 때, 대여 불가능 메세지 띄워주기
- 유저가 이미 책을 대여했을 경우, 안내 메세지 띄워주기
- 최신 댓글 순 정렬
- 댓글 평점 기능
- 댓글 내용과 평가 점수는 모두 필수로 받기
- 대여 기록, 반납 기록

-개인정보 보호조치 기준에 따른 비밀번호

-페이지네이션

-보기 좋은 화면, CSS

-비동기적 통신

-코드 리팩토링 (시간부족)

추가로 한 것

-재고가 없는 책을 예약할 수 있는 예약 기능

-본인이 작성한 댓글은 수정하거나 삭제할 수 있는 기능

-한 번에 대여 가능한 책을 두 권 까지로 제한
-마찬가지로 예약도 두 권 까지만 예약 가능하게 제한
-비슷한 로직으로 한 책에 예약을 3명까지로 제한
(A, B, C 유저가 예약을 걸었으면 더이상 예약을 받지 않음)

핵심 로직

Tables

-User : 유저 정보 (id, user_id, password, user_name)

-Book : 도서 정보 (id, title, publisher, author, pages, … , stock)

-Comment : 댓글 정보 (id, user_id, book_id, comment, time, rating)

-Rented : 대여 정보 (id, user_id, book_id, rented_time, returned_time)

-Reservation : 예약 정보 (id, user_id, book_id, isReserved)

배운 점

- 여러 기술적인 것들
- Wiki 작성은 선택이 아닌 필수이고 좋은 wiki가 짜임새 있는 프로젝트를 만드는 가이드 라인이 될 수 있다.
- Clean code를 짜는 법을 배워야만 한다... 내가 짠 코드도 못알아볼 지경...
- 오류 메세지는 문제 해결을 도와주는 우리들의 친구이다.

어려웠던 점 / 느낀점

Gitlab, Sqlalchemy, 배포 등 익숙하지 않아서 어려웠던점이 제일 많았습니다. 모르는 것들이 많았던 것도 어려웠습니다.

하지만 이렇게 프로젝트를 조그맣게나마 만들고 배포를 해 보니, 프로그램 생성과 배포에 정말 핵심적인 부분들을 많이 배웠다고 생각합니다. 앞으로 더 큰 프로젝트들은 그저 이것의 확장이라고 생각하니 전에 가지고 있던 막막함이 많이 해소가 됐습니다.

도움을 참 많이 받았는데, 코치님과 레이서 여러분들 감사합니다.

Q&A

로그인 / 회원가입

로그인 화면
유저를 입력받은 아이디로 검색해서,
있으면 세션에 추가
User.query.filter(User.user_id==id:입력창에서 받은 id:).first()
session['user_id'] = id

회원가입 화면
Db.session.add(User(id, pw_hash, name)
로그아웃 화면
session.clear()

도서관 책 목록

Book.query.order_by(Book.title.asc())

도서 상세 페이지

도서 목록에서

도서 상세 페이지

-book id를 url로 넘겨줘서

@board.route('/details/<int:book_id>')

과 같이 만들었습니다.

Book.query.filter(Book.id==book_id).first()

를 통해 도서 정보를 불러왔습니다.

도서 상세 페이지 / 대여

```
book=Book.query.filter(Book.id
==book_id).first()
book.stock-=1
rented=Rented(user_id, book_id)
db.session.add(rented)
db.session.commit()
```

도서 상세 페이지 / 대여

```
Rented.query.filter(Rented.book_id==id).filter(Rented.user_id==session.id).filter(Rented.returned_time==None).first()
```

한 유저가 빌린 한 책의 반납시간이 없다면, 이미 대여중인 책이므로 대여중이라는 메시지를 띄웁니다.

도서 상세 페이지 / 대여

Rented.query.filter(Rented.user_id == session.id).filter(Rented.returned_time == None).all()

한 유저가 대여중인 책들의 리스트입니다.
이 리스트에 len함수를 사용하여 대여 할 수 있는 책의 숫자를 제한했습니다.

도서 상세 페이지 / 예약

사실상 대여와 같은 로직입니다.

그러나 차이점은 한 유저가 예약한 한 책이 반납이 되면 예약한 순서대로 그 유저에게 자동적으로 대여가 되는 로직을 구성해 봤습니다. 마이 페이지에서 설명 하겠습니다.

도서 상세 페이지 / 댓글

Comment 테이블에 추가하는 단순한 로직입니다.

출력을 위해서 화면으로 보내준 comment객체에서 comment.id를 다시 서버로 보내고, 그 comment.id 와 Session.id로 query를 해서 자기가 쓴 글은 수정과 삭제를 가능하게 했습니다.

마이 페이지 / 반납

로그인된 유저가 빌린 책 (즉 반납시간이 None임) 을 화면에 보여줍니다.

반납을 하면 대여한 책의 stock을 1 늘리고 그 레코드에 반납시간을 추가해 줍니다.

예약 기능 (스무디 한잔 마시며 끝나는... test@test.com / test2@test.com)

만약 이것의 결과로 stock이 1이 되면, 그 뜻은 대기를 걸어놓은 유저가 빌려갈 책이 도서관으로 들어왔다는 뜻입니다. 그러므로 대기 1순위인 사람을 검색해서, 그 사람에게 도서를 대여해 줍니다.

(Reservation테이블에서 user_id 와 isReserved==True와 book_id인 제일 첫번째 값 (대기 1순위)를 찾아서 isReserved=False로 바꿔주고 (도서를 대여하므로 더 이상 예약 상태는 아니게 됩니다) 대여 로직을 실행합니다.)

마이페이지 / 대여기록

User_id, book_id, 반납시간 있음 으로 검색하여 화면에 나타냅니다.

여기서 책의 제목과 이미지를 보여주기 위해서 Book과 Returned를 페이지로 함께 보냈습니다.

이중 for문을 사용하여

Book.id와 Returned.book_id가 같은 경우에 페이지 (대여 목록) 를 보여주는 로직으로 나타냈습니다... 지금 생각난건데 이럴 때 join을 쓰면 좋을것 같습니다...