# Самостоятельная работа №1

## Техническое задание на разработку простейшего книжного магазина

В данном задании вам необходимо реализовать простейший сервис, поддерживающий `CRUD` для модели `Book`. Это означает, что пользователь (`User`) способен создавать/обновлять/удалять книги из магазина.

### Описание моделей

Заданы 2 модели:

```
class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(100), index=True, unique=True) # Имя
пользователя
    email = db.Column(db.String(100), index=True, unique=True) # Почта
пользователя
    password_hash = db.Column(db.String(128)) # Хеш пароля
    books = db.relationship('Book', backref='owner', lazy='dynamic') # Ссылка на
объекты модели Book

    ....
    ....

class Book(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100)) #  Название книги
    author = db.Column(db.String(100)) # Автор книги (тот, кто написал)
    pages_amount = db.Column(db.Integer) # Количество страниц
    published =  db.Column(db.DateTime, index=True, default=datetime.now) # Когда
появилась на сервсие (когда кто-то загрузил ее в магазин)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id')) # Собственник (тот,
кто загрузил на сервис)

    ....
    ....
```

Многоточия означают, что далее излагаются методы, необходимые для ваших моделей(`__str__`, `__repr__`, `set_password`, `check_password` и т.д.).

### Функционал

- `/register` - страница регистрации пользователя. Содержит на себе поля для ввода ***username, email, password, password_again*** и кнопку `Register`. В случае успешной регистрации (регистрация успешна когда ***username уникален***, ***email уникален и является действительно адресом эл.почты*** (см. валидатор `Email`), пароли совпадают). После регистрации - редирект на `/login` и `flash` уведомление `Account successfully created`. Если регистрация не успешна -

оставляем пользователя на `/register` и вывести соответствующие ошибки в виде `error_messages` рядом с полями формы регистрации + `flash` уведомление `Invalid credentials while register`.

- `/login` - страница пользовательского логина. Содержит на себе поля для ввода ***username*** и ***password***, также кнопку ***Log In***. После успешного логина (логин успешен - когда пользователь с таким ***username*** и ***password*** существует в бд), редирект на `/books` и `flash` уведомление `User successfully logged-in`. Если логин не успешен - оставляем пользователя на `/login` и вывести соответствующие ошибки в виде `error_messages` рядом с полями формы логина + `flash` уведомление `Invalid credentials while login`.

- `/logout` - страница пользовательского лог-аута. Содержит на себе кнопку `Log Out` и вопрос - `Confirm your log out as USERNAME`, где `USERNAME` - имя теущего пользователя, который залогинен в данный момент. Данный запрос необходимо закрыть декоратором `@login_required` (то есть не залогиненный пользователь не должен попадать на эту страницу). После `logout` редирект на `/`.

- `@login_required` декоратор отправляет всех пользователей, которые не залогинены на страницу `/login`

- `/` - домашняя страница, содержит на себе ссылки на `/books`, `/login` и `/register`.

- `/books` - список всех книг в произвольном порядке. Про каждую книгу должно быть выведено: `title, author, pages_amount, published, owner`. Поле `title` закрыто кликабельной ссылкой, отправляющей на страницу с детальной информацией. Запрос `/books` закрыт `@login_required` декоратором.

- `/books/<id>/detail` - страница с детальной информацией про книгу с `id`. Содержит на себе информацию `title, author, pages_amount, published, owner`. А также 3 кнопки `Edit` - для изменения информации про книгу с `id` (перенаправляет на страницу `/books/<id>/edit`), `Delete` - удаление книги с `id` (перенаправляет на страницу `/books/<id>/delete`), `To all Books` - возвращает на страницу `/books`. Запрос закрыт `@login_required` декоратором.

- `/books/<id>/edit` - страница редактирования книги с `id` содержит на себе форму с полями для изменения `title, author, pages_amount`, а также 2 кнопки `Edit` - подтверждает изменение информации и перенаправляет на страницу с детальной информацией про книгу с `id`, `Cancel` - возвращает пользователя на страницу с детальной информацией про книгу с `id` без отправки `POST` запроса. Ссылка закрыта `@login_required` декоратором.

- `/books/<id>/delete` - страница удаления книги с `id`. Содержит на себе строку `Confirm deleting this book` и кноку `Delete`. После нажатия на кнопку - книга удаляется из бд и происходит перенаправление пользователя на страницу `/books`. Ссылка закрыта `@login_required` декоратором.

- `/books/new` - страница создания новой книги. Содержит на себе форму для создания книги с полями `title, author, pages_amount`. Поле `owner` явное не определяется формой, а подставляется исходя из того, какой пользователь сейчас книгу создает. Так же на странице находится кнопка `Create` подтверждающая создание книги. После создания книги - перенаправление на страницу с детальной информацией про ***ЭТУ*** книгу. Ссылка закрыта `@login_required` декоратором.

## Решение

Ваше решение должно содержать в себе весь исходный код проекта, `.env` файл, `.db` файл бд, `pipfile и pipfile.lock`. Решение загрузить в `github` репозиторий и ссылку на него прислать преподавателю в чат.

## P.S.

Шаблоны (`.html`) можно использовать любые. То есть нет ограничение на присутствие/отсутствие `boostrap` панелей и т.д.