

Ruby on Rails를 이용한 웹페이지 제작

목 차

I . 서 론	1
1.1 서비스 주요 기능	1
1.2 메뉴 구조도	2
1.3 목표	2
II . 설계 수행 내용	5
1. 설계	5
1.1 주요화면 설계	5
1.2 DB 설계	10
1.3 flow-chart	11
2. 구현	12
III . 결 론	18
1. 시행착오	18
2. 향후 계획	18

1. 서론

오늘날 프리랜서가 증가하면서 이들을 위한 자리가 많아지고 있다. 그 예로 프리랜서와 이들의 전문적인 서비스를 원하고 필요로 하는 사람들을 매칭해주는 크몽이라는 웹페이지가 있다. 크몽은 유형의 상품을 판매/구매하는 것이 아니라 로고디자인, 번역, sns광고 등의 서비스를 판매/구매하는 페이지이다. 그래서 이 페이지를 참고하여 판매자와 구매자의 서비스를 필요로 하는 사람들을 매칭해주는 기능을 좀 더 간편하게 할 수 있도록 구현하고자 한다.

1.1 서비스 주요 기능

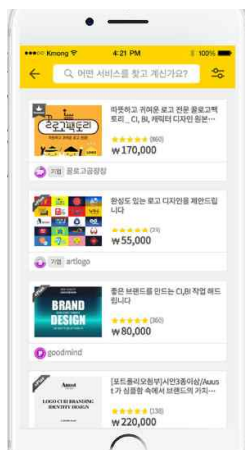
① 전문적인 서비스: 각 분야의 전문가들의 차별화된 서비스들을 이용할 수 있다.



<각 분야의 전문가>

② 간편한 구매 과정: 자신에게 맞는 서비스를 쉽게 찾아 간편하게 구매할 수 있다.

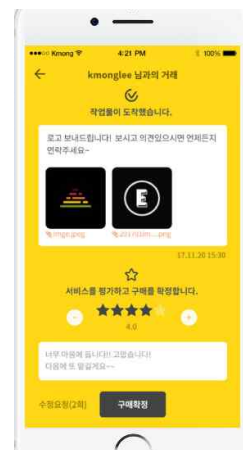
- 서비스 찾기 - 검색 또는 카테고리 선택을 통해 원하는 서비스들을 확인할 수 있다.
- 서비스 비교 - 서비스의 설명, 가격, 평가, 포트폴리오 등을 살펴, 견적에 맞게 옵션을 선택할 수 있다.
- 구매 및 평가 - 거래가 마무리되어 작업물을 받은 후, 전문가에 대한 평가를 남길 수 있다.



<a. 서비스 찾기>



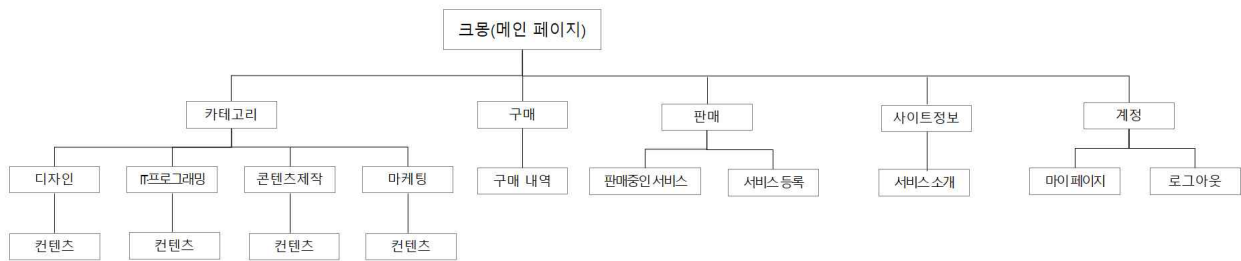
<b. 서비스 비교>



<c. 구매 및 평가>

③ 만족도 높은 결과물: 서비스를 찾아 비교할 수 있고, 서비스에 대한 평가 및 만족도를 확인할 수 있다. 그래서 정확하고 만족도 높은 서비스를 이용할 수 있다.

1.2 메뉴구조도



메뉴구조도는 크게 카테고리, 구매, 판매, 계정으로 구성했다. 원래 크몽의 카테고리는 총 11가지의 종류가 있지만, 여기에서는 4가지로 축소시켰다. 그리고 각 카테고리 안에는 그에 맞는 서비스 페이지가 있다. 다음으로 구매페이지에는 구매 내역을 볼 수 있고, 판매 페이지에는 판매중인 서비스와 서비스 등록페이지로 나눌 수 있다. 마지막으로 계정에는 마이페이지와 로그아웃이 있는데 마이페이지는 구매 및 판매와도 다시 연결된다.

1.3 목표

조사한 서비스의 주요기능과 작성한 웹페이지의 메뉴구조도를 바탕으로 주요화면과 DB를 설계한다. 먼저 rails new 명령으로 디렉토리를 만들어 어플리케이션을 생성한다. 그리고 이 폴더로 이동해 rails g scaffold 명령을 통해 필요한 MVC를 생성한 후, 모델에서 각 테이블 간의 관계를 설정한다.

기능 구현을 위해 필요한 메소드들은 컨트롤러와 모델에 정의한다. 그리고 컨트롤러와 뷰를 수정하며 각 페이지의 기능을 설정하고, 페이지 간 이동이 쉽도록 구현한다.

[표 1] 웹사이트의 구성요소

설계 구성요소	내용
합성	<ul style="list-style-type: none"> - 프리랜서들의 재능 판매를 위한 플랫폼 개발 - 재능이나 기술을 필요로 하는 사람들이 서비스를 이용할 수 있는 것이 목표 - 각 페이지 간에 이동이 원활하고, 모든 기능들이 데이터베이스와 관련하여 잘 작동하는 것이 평가 기준 - 유사한 서비스를 제공하는 웹 사이트를 조사하여, 각 팀원들이 목적에 맞게 구현
분석	<ul style="list-style-type: none"> - 유사한 서비스를 제공하는 '크몽'사이트를 참조, 필요한 요소들의 데이터베이스와 메뉴 구조도 설계 - 접속한 사용자의 입장에서, 서비스 구매나 판매, 그리고 서비스 관리 기능을 이해 - 웹 개발 도구인 루비온레일즈 프레임워크를, sqlite 데이터베이스를 사용 - 각 데이터베이스 테이블 간에 연결 문제 예상
제작	<ul style="list-style-type: none"> - 각 요소들의 데이터베이스 테이블 추가 Category 테이블, User 테이블, Service 테이블, User와 Service의 조인 테이블인 User_Service 테이블 - 각 요소들을 통합한 전체 시스템의 구현 회원가입 및 로그인, 카테고리 속 서비스들의 목록 확인과 구매, 구매내역 보기 기능, 그리고 판매와 판매내역 확인
시험	<ul style="list-style-type: none"> - 각 기능들의 원활한 동작 확인 - 전체 시스템의 동작 확인 - 서비스 판매를 목적으로 할 때와 구매를 목적으로 할 때의 흐름으로 동작 확인 - 오동작 발생 시 문제 파악 및 코드수정 후, 재확인함
평가	<ul style="list-style-type: none"> - 사용자의 입장에서 요구 사항의 충족 여부 평가 - 충족하지 못한 요구 사항에 대한 분석 - 수정 및 보완 사항 정리

[표 2] 설계 웹페이지의 제한요소

설계 제한요소	내용
안정성	<ul style="list-style-type: none"> - 설계 목표에 부합하는 동작 - 의도되지 않은 입력이나 잡음에 대한 오류 최소화
신뢰성	<ul style="list-style-type: none"> - 시스템 출력의 낮은 오류 - 시스템의 합리적인 반응 속도 - 동일한 입력에 대한 동일한 출력 보장 - 시스템 조작자와 무관한 동작 결과
미학	<ul style="list-style-type: none"> - 페이지 간 이동의 용이성 - 웹 페이지의 디자인

2. 설계 수행 내용

1. 설계

1.1 주요화면 설계

① 로그인 : ID와 비밀번호 또는 다른 계정으로 로그인 가능하다.

사이트명 또는 로고

로그인

아이디

비밀번호

로그인

아이디/비밀번호 찾기 회원가입

페이스북으로 시작하기

네이버로 시작하기

카카오톡으로 시작하기

구글로 시작하기

② 회원가입 : 필수적으로 ID, 비밀번호, 닉네임을 입력해야하고, 선택적으로
연락가능시간, 전화번호, 경력사항, 이미지 등을 입력할 수 있다.

사이트명 또는 로고 내용 입력해주세요 로그인 **회원가입**

디자인 IT · 프로그래밍 콘텐츠 제작 마케팅

회원가입

대표이미지 설정

아이디 비밀번호

내용 입력해주세요 내용 입력해주세요

닉네임

내용 입력해주세요

연락가능시간 전화번호

~ | 내용 입력해주세요

경력사항

수상경력

제작자정보 제작자 : 0000 전화번호 : 0000-0000 이메일 : 0000@0000	카테고리 디자인 IT · 프로그래밍 콘텐츠 제작 마케팅	사이트 정보 서비스 소개 공지사항 FAQ 이용약관
---	---	--

③ 메인화면 : 카테고리별로 인기 서비스를 보여준다.

사이트명 또는 로고

내용을 입력해주세요

로그인

회원가입

디자인

IT · 프로그래밍

콘텐츠 제작

마케팅

디자인 카테고리에서 인기 있어요!

IT · 프로그래밍 카테고리에서 인기 있어요!

콘텐츠 제작 카테고리에서 인기 있어요!

마케팅 카테고리에서 인기 있어요!

제작자 정보

카테고리

사이트 정보

제작자 : □□□

서비스 소개

전화번호 : □□□-□□□□

공지사항

이메일 : □□□@□□□□

콘텐츠 제작

FAQ

마케팅

이용약관

④ 카테고리 : 각 카테고리별로 서비스들을 보여준다.

사이트명 또는 로고

내용을 입력해주세요

로그인

회원가입

디자인

IT · 프로그래밍

콘텐츠 제작

마케팅

카테고리명

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

컨텐츠 명

가격 : XXX

<

1

2

3

4

>

제작자정보

카테고리

사이트 정보

제작자 : □□□
전화번호 : □□□.□□□
이메일 : □□□@□□□

디자인
IT · 프로그래밍
콘텐츠 제작
마케팅

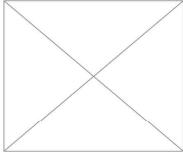
서비스 소개
공지사항
FAQ
이용약관

- ⑤ MY Page : 정보수정을 할 수 있는 페이지로 가는 버튼이 있고,
구매/판매내역과 판매중인 서비스들을 확인 할 수 있다.

사이트명 또는 로고
로그인
회원가입

디자인
IT · 프로그래밍
콘텐츠 제작
마케팅

내 정보



회원정보 수정

총 구매 개수

총 판매 개수

구매내역

컨텐츠명	구매가격	판매자	구매날짜

< 1 2 3 4 >

판매내역

컨텐츠명	판매가격	판매 개수	판매(시작)날짜

< 1 2 3 4 >

제작자정보

제작자 : □□□
전화번호 : □□□-□□□□
이메일 : □□□@□□□

카테고리

디자인
IT · 프로그래밍
콘텐츠 제작
마케팅

사이트 정보

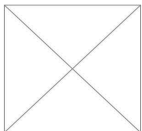
서비스 소개
공지사항
FAQ
이용약관

- ⑥ 정보수정 : 이메일, 닉네임, 연락가능시간, 이미지 등의 수정이 가능하다.

사이트명 또는 로고
로그인
회원가입

디자인
IT · 프로그래밍
콘텐츠 제작
마케팅

계정 설정



이미지 수정

이메일

닉네임

연락가능시간

 ~

전화번호

취소
수정하기

제작자정보

제작자 : □□□
전화번호 : □□□-□□□□
이메일 : □□□@□□□

카테고리

디자인
IT · 프로그래밍
콘텐츠 제작
마케팅

사이트 정보

서비스 소개
공지사항
FAQ
이용약관

⑦ 서비스 등록 : 판매할 서비스에 대한 정보를 구체적으로 입력한다.
(이미지, 제목, 작업기간, 카테고리, 서비스 내용 등)

사이트명 또는 로고

로그인회원가입

디자인

IT · 프로그래밍

콘텐츠 제작

마케팅

컨텐츠 등록

서비스 제목

내용을 입력해주세요요

카테고리 설정

카테고리

작업기간
수정될 수

작업기간

수정될 수

작업내용 제공

☒ 제공☐ 미제공

가격 설정

내용을 입력해주세요요

원

서비스 설명

내용을 입력해주세요요

취소 · 환불 규

내용을 입력해주세요요

대표이미지 설정

취소

등록하기

제작자정보

카테고리

사이트 정보

제작자 : 김민서
전화번호 : 010-0000-0000
이메일 : k.m.s@naver.com

디자인
IT · 프로그래밍
콘텐츠 제작
마케팅

서비스 소개
공지사항
FAQ
이용약관

⑧ 서비스 : 판매자가 등록한 서비스와 그 내용을 보고 구매할 수 있다.

사이트명 또는 로고

내용을 입력해주세요

로그인회원가입

디자인IT · 프로그래밍콘텐츠 제작마케팅

홈 > 카테고리명

서비스 제목
가격 : 0000

구매하기

서비스 설명

서비스 설명
서비스에 대한 설명
서비스에 대한 설명
서비스에 대한 설명
서비스에 대한 설명
서비스에 대한 설명

가격 정보

가격 : 0000 원
작업기간 : 0 일
수정횟수 : 0 번
작업내용 제공 유무 : ☒ 예 ☐ 아니요

취소·환불 규정

취소 환불 규정
취소 환불 규정
취소 환불 규정
취소 환불 규정
취소 환불 규정
취소 환불 규정

사용자 평가

사용자 별
사용자 평가 내용
사용자 평가 내용

사용자 별
사용자 평가 내용
사용자 평가 내용

사용자 별
사용자 평가 내용
사용자 평가 내용

< 1 2 3 4 >

제작자 정보

제작자 : 0000
전화번호 : 00 00 00 00 00
이메일 : 00 00@00 00.co.kr

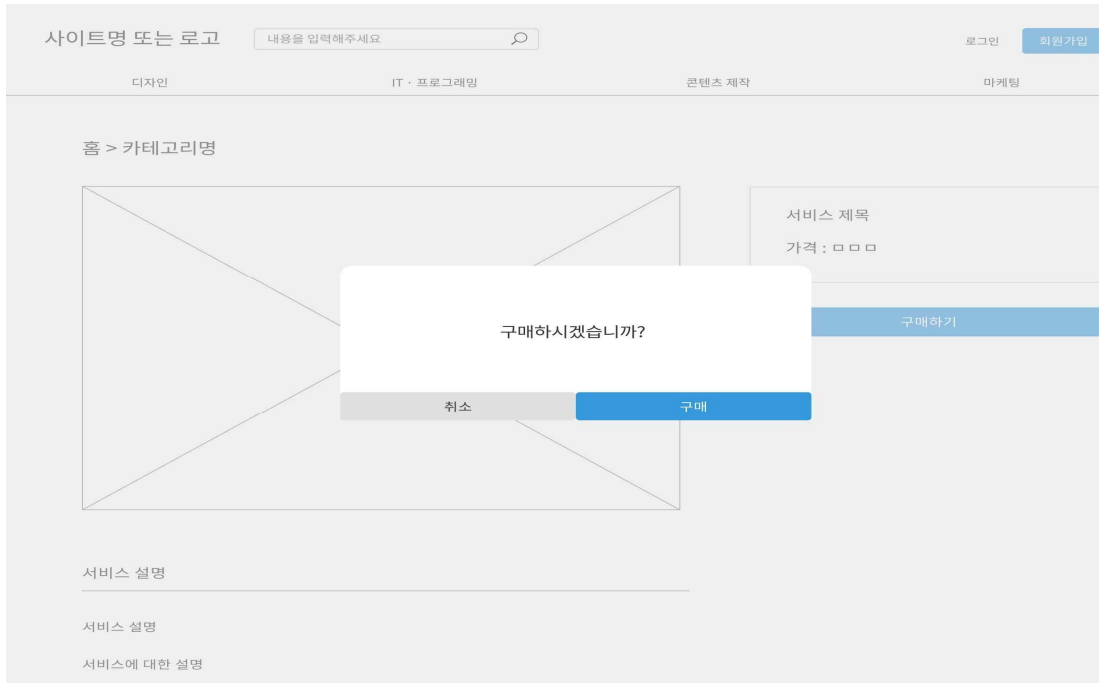
카테고리

디자인
IT · 프로그래밍
콘텐츠 제작
마케팅

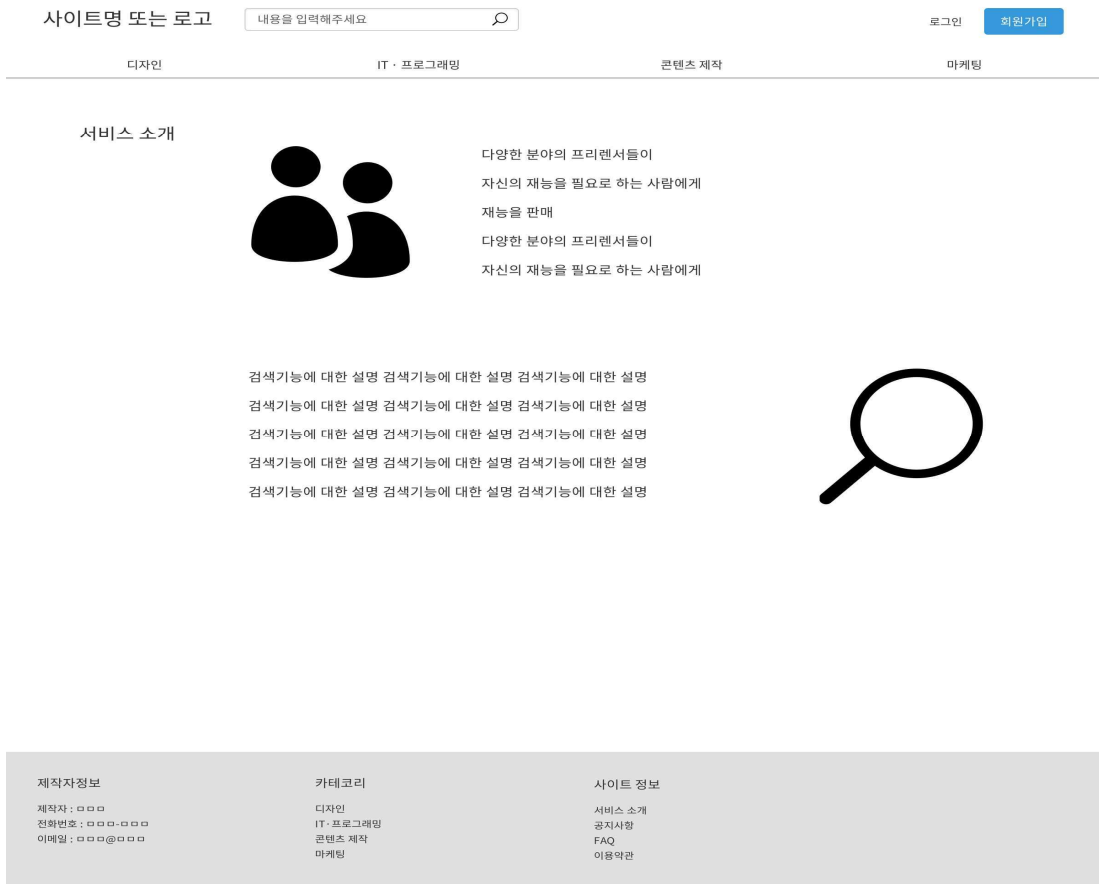
사이트 정보

서비스 소개
공지사항
FAQ
이용약관

⑨ 구매 : 구매를 할 것인지 확인하는 창이다.



⑩ 서비스 소개 : 이 웹페이지에 대한 소개 글이다.



⑪ 후기 : 구매한 서비스에 대한 후기를 작성할 수 있는 페이지이다.

사이트명 또는 로고

내용을 입력해주세요

로그인

회원가입

디자인

IT · 프로그래밍

콘텐츠 제작

마케팅

후기 작성

제목

내용을 입력해주세요

평점

내용

내용을 입력해주세요

취소

평가완료

제작자정보

제작자 : □□□

전화번호 : □□□-□□□□

이메일 : □□□@□□□

카테고리

디자인

IT · 프로그래밍

콘텐츠 제작

마케팅

사이트 정보

서비스 소개

공지사항

FAQ

이용약관

1.2 DB 설계

```

    erDiagram
        User ||--o{ User_Service : "has"
        Service ||--o{ User_Service : "has"
        Category ||--o{ Service : "has"

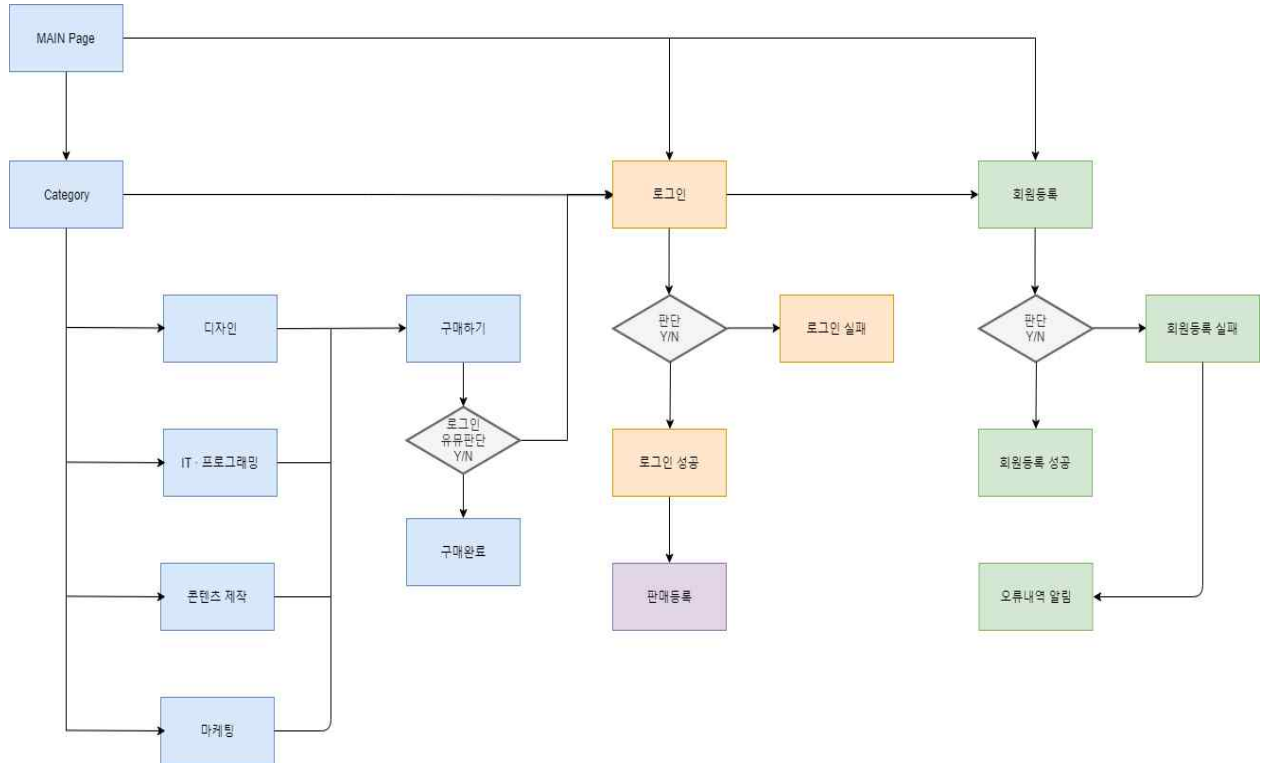
        User {
            int id PK
            varchar(45) name
            varchar(45) email
            varchar(45) phone
            text address
            int age
        }
        Service {
            int id PK
            varchar(45) name
            int price
            int category_id FK
            int user_id FK
        }
        Category {
            int id PK
            varchar(45) name
        }
        User_Service {
            int user_id FK PK
            int service_id FK PK
        }
    
```

총 4개의 테이블을 생성할 예정이며 각 테이블의 용도는 다음과 같다.

User테이블은 회원가입 당시 사용자의 정보를 저장하기 위해, Service 테이블은 판매할 상품의 정보를 저장하기 위해 생성 되었다. Category 테이블은 서비스의 카테고리 분류를 위해 존재하며 관리자만 사용가능하기에 일반 사용자가 카테고리 테이블에 접근하여 생성 또는 수정할 일은 없다. 마지막으로 UserService 테이블은 회원으로 등록한 사용자가 서비스를 구입하면 그 내용이 저장되는 테이블로써 추후 마이페이지의 구매내역에 그 내용이 표시된다.

- 10 -

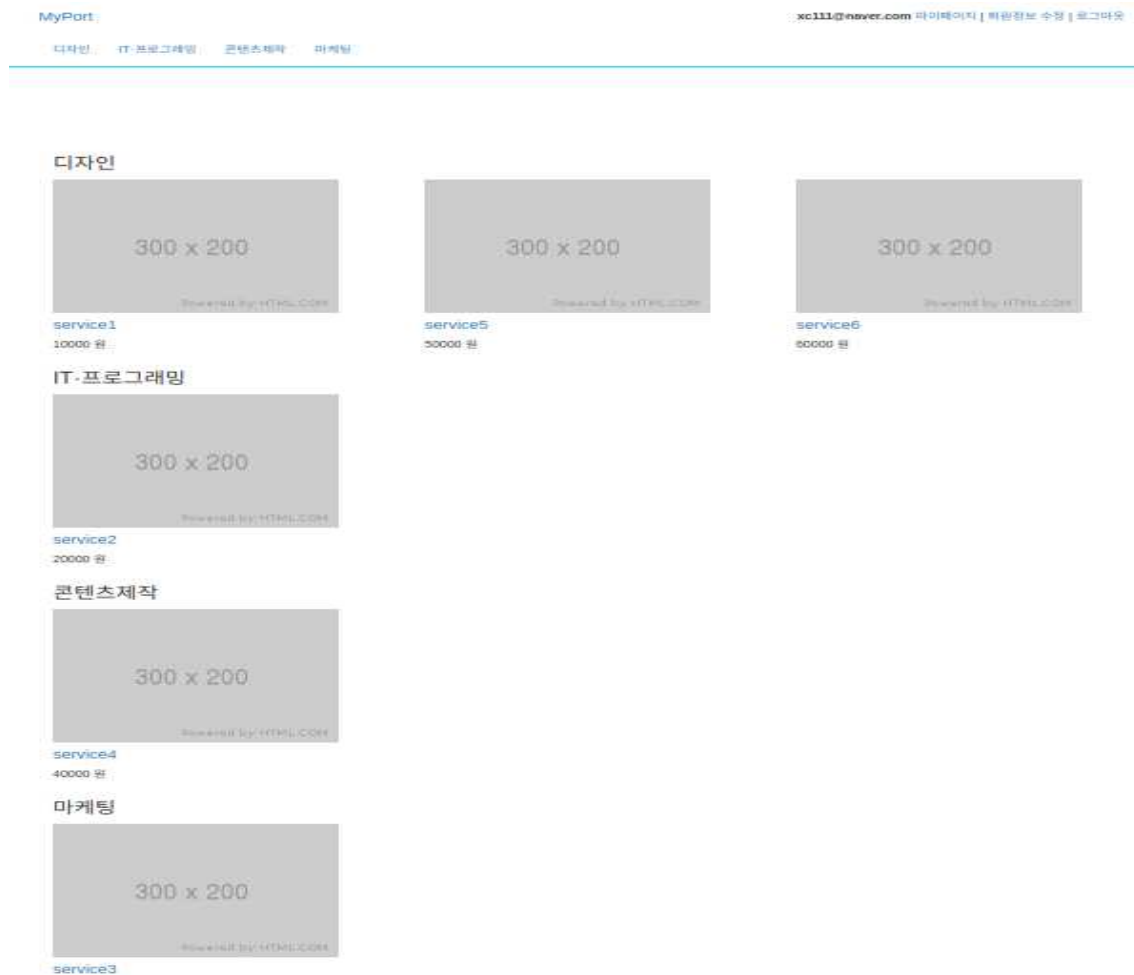
1.3 flow-chart



메인 페이지에서 카테고리, 로그인, 회원등록으로 이동할 수 있다. 카테고리에는 디자인, IT, 콘텐츠, 마케팅 페이지가 있으며 이 페이지에서 서비스 상세 페이지로 이동하여 구매할 수 있다. 만약 구매 시 로그인이 되어있지 않으면 로그인 페이지로 이동한다.

로그인 페이지에서는 email과 password를 입력하여 로그인을 성공하면 서비스를 구매, 판매할 수 있다. 그리고 실패 시에는 회원가입 페이지로 이동하여 회원가입을 할 수 있다. 그래서 회원가입이 성공하면 바로 로그인이 된 메인 페이지로 이동하고, 안되면 오류 알림이 뜬다.

① 메인페이지



<app/views/main/index.html.erb>

```
</head>
<body>
  <div class="container top-padding">
    <% @category.each do |category| %>
      <div class="container">
        <h3 class="my-4"><%= category.category_name%></h3>
        <div class="row">
          <% category.cs_list.each do |service| %>
            <% if service.category_id == category.id %>
              <div class="col-lg-4 col-sm-6 mb-4">
                <div class="card h-100">
                  <a href="#"></a>
                  <div class="card-body">
                    <h4 class="card-title"><%= link_to service.service_name,
service_path(service.id) %></h4>
                    <p class="card-text"><%= service.price %> 원</p>

```

메인페이지를 보여주기 위해 main 컨트롤러와 뷰를 생성하였고 뷰에 들어갈 내용은 위와 같다.

각 카테고리별로 DB에 저장되어있는 서비스를 보여주기 위해 반복자를 중첩시켜서 사용했다.

<app/models/category.rb>

```
1 class Category < ApplicationRecord
2   has_many :services
3
4   def cs_list
5     return Service.where(category_id: self.id).all.limit(3)
6   end
7 end
8
```

메인페이지의 카테고리별 서비스를 3개만 보이도록 제한을 걸어두는 메소드로 category 모델에 선언되어 있다.

이 메소드는 현재 객체(category)의 id값을 Service의 category_id와 비교해서 두 id가 동일한 행들을 배열형태로 되돌려주며 DB에 저장된 순서대로 최대 3개까지만 저장한다.

② 카테고리 페이지

MyPort

xc111@naver.com 마이페이지 | 회원정보 수정 | 로그아웃

디자인 IT-프로그래밍 콘텐츠제작 마케팅

디자인

140 x 100	140 x 100	140 x 100	140 x 100
service1	service5	service6	service7
가격 : 10000 원	가격 : 50000 원	가격 : 60000 원	가격 : 70000 원
판매자 : xc111@naver.com	판매자 : xc111@naver.com	판매자 : xc111@naver.com	판매자 : xc111@naver.com

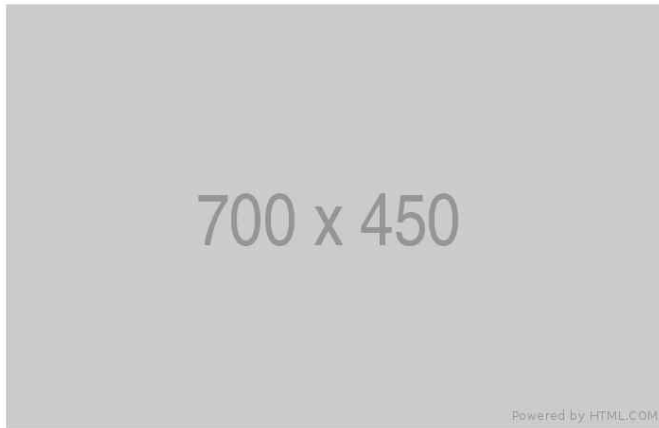
각 카테고리의 페이지는 등록되어있는 서비스를 한 행에 4개씩 보여주며 서비스 등록 당시 첨부한 대표 이미지를 보여주고 서비스의 제목과 가격, 판매자의 이메일을 나타내 준다. 또한, 서비스의 제목을 클릭하면 해당 서비스의 상세내용을 볼 수 있는 페이지로 넘어간다.

③ 서비스 상세 페이지

MyPort

xc111@naver.com 마이페이지 | 회원정보 수정 | 로그아웃

디자인 IT·프로그래밍 콘텐츠제작 마케팅



service1

가격 : 10000 원

카테고리 : 디자인

판매자 : xc111@naver.com

구매하기

서비스후기

해당 서비스의 상세페이지 접속 시 보이는 화면으로 서비스의 대표 이미지를 좌측에 보여주고 우측에는 서비스의 간단 정보와 구매하기 버튼을 구현해 두었다.

구매하기 버튼은 로그인 하지 않은 사용자가 누를 시 로그인 화면으로 가도록 설정해 두었다.

<app/views/services/_form.html.erb>

```
<p>
  <%= if user_signed_in? %>
    <%= button_to '구매하기', buy_service_path(current_user.id, @service.id),
      method: :post %>
  <%= else %>
    <%= button_to '구매하기', new_user_session_path %>
  <%= end %>
</p>
```

앞서 설명한 바와 같이 구매하기 버튼을 누를 때 로그인 하지 않은 사용자는 로그인 페이지로 가도록 설정한 코드이며 메소드로 구현하지 않고 HTML단에서 if, else문을 통해 동작하도록 해 두었다.

④ 로그인 및 회원가입

MyPort

디자인

IT·프로그래밍

콘텐츠제작

마케팅

회원가입

이메일

비밀번호

비밀번호확인

이름

나이

전화번호

주소

가입하기

MyPort

디자인

IT·프로그래밍

콘텐츠제작

마케팅

로그인

이메일

비밀번호

☐ 계정정보기억하기

로그인

[회원가입](#)

[비밀번호 찾기](#)

로그인페이지와 회원가입 페이지는 devise를 통해 생성했으며 form에 적혀있던 영문을 한글로 수정했다.

⑤ 마이페이지

MyPort

xc111@naver.com 마이페이지 | 회원정보 수정 | 로그아웃

디자인 IT-프로그래밍 콘텐츠제작 마케팅

마이페이지

서비스 생성

구매개수 2
판매중인 상품 개수 5

구매목록

User	Service	가격
2	service3	30000
2	service3	30000

판매목록

서비스 이름	서비스 가격	판매 개수	
service2	20000	3	Destroy
service1	10000	0	Destroy
service5	50000	0	Destroy
service6	60000	0	Destroy
service7	70000	0	Destroy

마이페이지에서는 자신이 구매 또는 판매한 목록을 보여주며 서비스 생성이 가능하다. 또한 총 구매개수와 판매개수를 상단에 나타내 준다.

<app/models/user_service.rb>

```

1  class UserService < ApplicationRecord
2    belongs_to :user, required: false
3    belongs_to :service, required: false
4
5    def returnService
6      return Service.find_by_id(self.service_id)
7    end
8
9    def Service_name
10     return Service.find_by_id(self.service_id).service_name
11   end
12
13   def Service_price
14     return Service.find_by_id(self.service_id).price
15   end
16
17   def seller_name
18     service = Service.find_by_id(self.service_id)
19     seller_email = User.find_by_id(service.user_id).email
20     return seller_email
21   end
22 end
23

```

위의 코드들은 user_service모델에 정의된 메소드로서 5행부터 15행까지의 내용은 해당 서비스의 칼럼 값을 넘겨주는 동작으로 기능이 동일하다.

17행부터 21행의 메소드는 구매목록에 판매자의 이메일 값을 불러오기 위해 생성했는데 우선 service_id를 통해 어떤 서비스인지 알려주기 위해 18행과 같이 선언해 두었다. 이후 19행에서 User에 service의 user_id값을 넘겨서 해당 유저가 누구인지 알아내고 그 유저의 email을 return시키도록 했다.

⑥ 서비스 생성 페이지

MyPort

디자인

IT·프로그래밍

콘텐츠제작

마케팅

서비스 생성

서비스이름

가격

카테고리 디자인 ▼

Create Service

[돌아가기](#)

서비스 생성페이지는 로그인한 사용자만 사용이 가능하며, 생성할 서비스의 이름과 가격을 명시한 뒤 카테고리는 리스트박스를 통해 선택하면 db에는 자동적으로 해당 서비스를 생성한 유저의 ID값과 카테고리의 ID값이 저장된다.

```
<div class="field">
  <%= form.label :카테고리 %>
  <%= form.select(:category_id) do %>
    <%= options_from_collection_for_select(Category.all, :id, :category_name) %>
  <% end %>
</div>
```

서비스 생성 시 카테고리를 직접 적어서 표시하는 것 보다 리스트 박스를 통해 선택하는 것이 사용자 측면에서 편할 것으로 생각되어 service 생성 form에 category를 적는 부분을 textfield형태에서 select형식으로 변경하였다.

```
def create
  @service = Service.new(service_params)
  @service.user_id = current_user.id
  respond_to do |format|
    if @service.save
      format.html { redirect_to buy_list_path(current_user.id) }
      format.json { render :show, status: :created, location: @service }
    else
      format.html { render :new }
      format.json { render json: @service.errors, status: :unprocessable_entity }
    end
  end
end
```

이 또한 서비스 생성자의 ID값을 생성자가 직접 명시할 것이 아니라 현재 접속해 있는 유저의 ID값을 서비스 생성 시 DB에 자동적으로 저장될 수 있도록

@service.user_id = current_user.id 라는 코드를 추가해 주었다.

3. 결론

1. 시행착오

ruby on rails라는 프레임워크를 사용하면서 가장 어려움을 느꼈던 부분은 DB의 연결 관계와 MVC아키텍처에 대한 이해였다. MVC아키텍처에서 가장 중요한 것은 뷰를 통해 사용자가 action을 취하면 컨트롤러에서 받아들이고, 이 요청에 대해 DB에서 값을 불러 오거나 저장을 한다. 그리고 결과를 다시 View를 통해 사용자에게 보여주는 것이다.

결국은 컨트롤러와 모델을 통한 DB의 입출력을 자유자재로 다룰 수 있어야 모든 기능들을 구현할 수 있다. 그런데 DB에 대한 선행 지식이 없었기 때문에 정말 힘들었지만 약 3주 동안 밤을 새워가며 수많은 오류를 처리하며 문제들을 파악하게 되었고, 쿼리문 또한 자연스럽게 사용법을 익히게 되었다.

로그인을 안 했을 때 메인페이지가 로그인 화면으로 뜨고 서비스들을 볼 수 없는 문제가 있었다. 그래서 컨트롤러에 before_action :authenticate_user! 삭제하여 해결하였다. 그리고 서비스를 등록할 때 설정된 카테고리가 아닌 다른 카테고리를 작성하면 안 된다. 그래서 지정된 카테고리 중에서 선택하도록 옵션태그를 생성하여 해결했다.

비회원이 서비스를 생성하거나 등록되어있는 서비스를 수정, 삭제할 문제를 방지하기 위해 if문을 사용해 해결했다. 또한, 회원 본인이 등록하지 않은 서비스를 수정, 삭제하는 문제와 본인이 등록한 서비스를 구매하는 문제를 해결하기 위해 if문을 사용했다.

2. 향후 계획

설계한 주요기능 중 후기기능을 구현하기 위해서는 우선적으로 후기를 저장하기 위한 전용 테이블을 생성할 필요가 있다. 그 테이블의 칼럼에는 후기를 작성하는 사용자의 ID값과 해당 서비스의 ID값을 저장할 칼럼이 있고 후기를 저장할 Text라는 칼럼이 있어야 한다. 후기는 별도의 페이지를 두지 않고 서비스의 상세 정보 페이지 하단에 작성할 수 있도록 Textfield를 만들 것이다.

[표 3] 설계 목표의 중요도 및 달성도

목표	중요도(%)	달성도(%)	수행내용
주요화면설계	10	100	주요화면을 설계함
DB설계	20	100	DB를 설계하고, 각 관계를 정의함
flow-chart	10	100	화면이동에 맞게 flow-chart를 구성
기능 구현	60	80	각 기능 구현함(로그인, 구매, 판매)
합계	100		