



Michelin in TV

TV에서 만나는 미슐랭 식당

강내원
20181669 이우진
20191624 이승우



목차

1. 프로젝트 소개
2. 역할 분담
3. 시연
4. 개발 이슈



프로젝트 소개

개발 배경

- Netflix에서 방영한 **흑백요리사**
- 전국적으로 미슐랭 식당의 인기
- TV에서 미슐랭 식당 정보를 얻을 수 있다면..?



"이렇게 잘될 줄은"...흑백요리사 열풍 외식업계 들쭉

등록 2024.10.07 12:28:44 수정 2024.10.07 16:50:44

📧 📷 📄 📌 📁



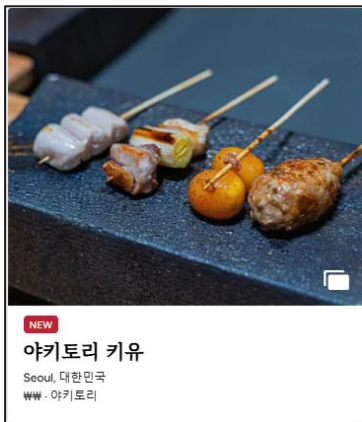
[서울=뉴스1] 김혜진 기자 = 흑백요리사에 출연 중인 최현석(왼쪽부터), 정지신, 장호준, 에드워드 리가 7일 오전 서울 마포구 호텔 나루 서울 앵겔러에서 열린 넷플릭스 예능 '흑백요리사: 요리 계급 전쟁' 기자회견에서 포즈를 취하고 있다.

2024.10.07. jini@news1.com

프로젝트 개요

- 미술랭 식당을 소개하는 앱
- 사용자는 소개 영상을 통해 식당의 분위기, 메뉴 등을 알 수 있음

이름	위치
소개 영상	가격대
미술랭 스타	소개 글





진행 과정



프로젝트 설계

- 프로젝트 주제 선정
 - 미슐랭 식당 가이드
- 역할 분담
- 규칙
 - 매주 화요일 2시에 회의 및 진행상황 공유

인원	역할
강내원	백엔드, DB
이우진	설계문서 작성, enact
이승우	enact

문서 작업

- API 문서 작성 및 mock API 생성

Michhelin Backend REST API Reference

1. 키워드 기반 레스토랑 추천목록 가져오기

키워드 기반 레스토랑 추천목록 가져오기는 backend 서버의 데이터베이스에 저장되어 있는 레스토랑의 id, restaurant_name, rating, city, img를 가져오는 API입니다.

웹 애플리케이션의 메인 화면에서 frontend에서 backend 서버의 키워드 기반 레스토랑 추천목록 가져오기 API를 호출하면 데이터베이스에 저장된 레스토랑 중 랜덤하게 선택한 키워드를 기반으로 분류된 레스토랑들의 목록을 응답합니다.

Request

ID	URL	HOST	METHOD
	/restaurants/recommendation		GET

Response

Name	Type	Description
type	String	restaurant의 종류
restaurants	Object[]	restaurant_id, restaurant_name, rating, city, img
restaurant_id	ObjectId	restaurant document의 ObjectId
restaurant_name	String	restaurant의 이름
rating	Number	restaurant의 미슐랭 등급
city	String	restaurant이 위치한 도시
img	String	restaurant의 이미지 주소

<docs/REST-API.md>

```
export const mockAPI = async (url, method, parameters = {}) => {
  const response = {
    '/restaurant/recommendations': {
      // keyword에 해당하는 레스토랑들을 불러온다.
      GET: [
        {
          type: 'Italian',
          restaurants: sampleRestaurants,
        },
        {
          type: 'Chinese',
          restaurants: sampleRestaurants,
        },
        {
          type: 'Korean',
          restaurants: sampleRestaurants,
        },
      ],
    },
  },
}
```

<frontend/_mocks_/api/request>

문서작업

- 설계 문서 작성
- 프로젝트 목적
- 요구사항
- 테스트

Michelin in TV 요구사항 명세서

목차

[1. 문서 설명\(Document Description\)](#)

[1.1. 목적\(Purpose\)](#)

[1.2. 범위\(Scope\)](#)

[1.3. 용어 및 정의\(Terminologies and Definitions\)](#)

[2. 시스템 컨텍스트\(System Context\)](#)

[2.1. 시스템 인터페이스\(System Interface\)](#)

[2.2. 시스템 컨텍스트\(System Context\)](#)

[2.3. 사용자 인터페이스\(User Interface\)](#)

[3. 상세 요구사항\(Specific Requirements\)](#)

[3.1. 기능 요구사항\(Functional Requirements\)](#)

[3.2. 품질 요구사항\(Quality Attributes\)](#)

[3.3. 제약 사항\(Constraint Requirements\)](#)



본격적인 개발

- 문서를 참고해서 동시에 개발
- 매주 화요일 진행 현황 공유 및 테스트 (시스템 통합)
- 과제 할당

테스트

- 전주에 작성한 테스트 문서 기반 평가

Backend Unit Test Cases

Backend API 검증은 postman을 활용한 unit test를 사용한다.

API	Test Name	Test Case ID	Description	Test Data
/restaurant/recommendations	키워드 기반 레스토랑 추천 조회	TC01-1	<ul style="list-style-type: none">Status code 200: keyword로 레스토랑 목록 조회 성공 시 응답 타입 및 필드 타입 검증Status code 500: Internal server error 메시지 검증	GET /restaurant/recommendations?keyword=H

<docs/test_plan.md>

Frontend System Test Cases

Frontend의 사용자 interaction과 view는 use case를 기반으로한 system test를 수행한다.

1. 레스토랑 검색 시나리오 (TC10-1)

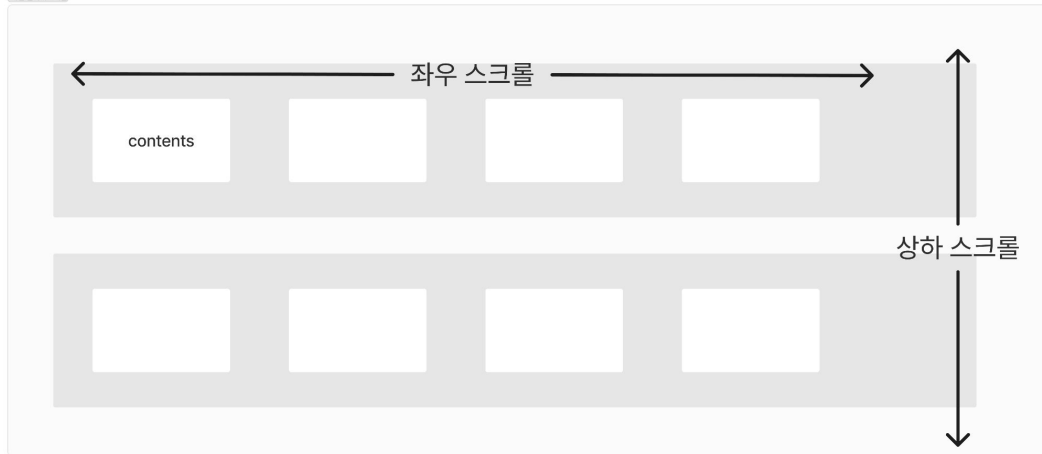
Test Step	Test Data	Expected Result
사용자가 TV에서 Michelin in TV 앱을 실행한다.		앱 메인 화면에 검색 인터페이스와 추천 레스토랑 목록 표시
사용자가 검색 인터페이스에 국밥 라고 입력한다.	"국밥"	"국밥" 키워드를 포함한 레스토랑 목록 표시
검색 결과가 없을 경우 "XYZQW" 입력	"XYZQW"	"결과가 없습니다" 메시지 표시

<docs/test_plan.md>

테스트 - 메인 패널

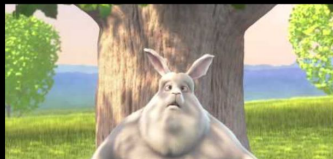
요구 사항: 상하 스크롤, 좌우 스크롤이 가능해야 한다. (Netflix UI 참고)

외면 설계서 - 메인



테스트 - 메인 패널

Italian



La Trattoria
Rome 🍷🍷🍷



Osteria del Corso
Florence 🍷



Ristorante Napoli
Naples

Chinese






결과

메인 페이지


MICHELIN in TV

🔍 👤 🌐 ↶ ✕


돼지국밥




합천국밥집
Busan, 대한민국



나막집
Busan, 대한민국







안암
Seoul, 대한민국



옥동식
Seoul, 대한민국

한식



Info 페이지



Info 페이지

MICHELIN in TV

좋아요

별로예요

방문

합천국밥집

남구 유호로 235 Rusan 4R5Q3 대하미국

부산 미술관 선정된 용호동 ...

3:22

cpu
23.08%

memory
40.53%

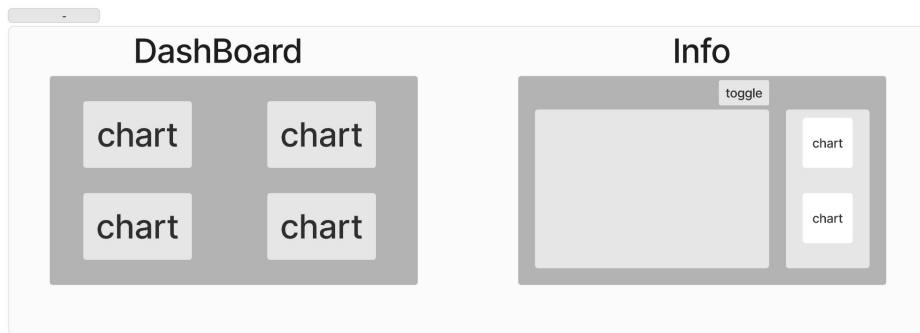
Network
167.77 Mbps



개발 이슈

대시보드

- 요구사항
 - 자원 현황 (cpu, memory, network speed) 를 시각화된 자료로 관찰
 - DashBoard panel, Info - 두 패널에서 확인할 수 있어야 한다.





대시보드

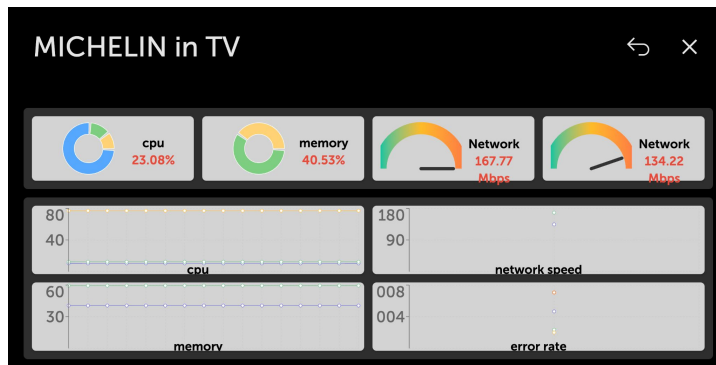
cpu 사용률, 메모리 사용률, 네트워크 속도 → 동영상 재생 시 중요한 **metric**

- CPU, Memory 사용량 → **template project** 참고해서 사용률 추출
- 네트워크 속도 → **webosose** 참고해서 **connectionmanager/monitorActivity** 호출
 - wifi 관련 데이터 추출
 - rxBytes, txBytes, txErrors, rxErrors, txDropped, rxDropped, txPackets, rxPackets
 - 문서에 **reference** 기반으로 **mock data** 생성

<https://www.webosose.org/docs/reference/ls2-api/com-webos-service-connectionmanager/#monitoractivity>

대시보드

- recharts 활용
 - react 와 호환성
 - 자세한 문서 제공
 - pie chart (cpu, memory), line chart (시간에 따른 경향성)
- svg를 활용한 게이지 차트
 - 네트워크 속도





대시보드 - 실시간 업데이트

- 디버깅
 - useProcStat 은 계속 데이터 출력
 - DashBoard까지 전달이 안됨
- 복잡한 구조 문제?
 - DashBoard → useSystemStatistics → useProcStat → mem → request → luna api
 - DashBoard → useProcStat → mem → request → luna api



대시보드 - 실시간 업데이트

- 디버깅
 - 구독이 제대로 이루어지지 않음



마무리

- 문서 작성을 통해 팀원 간 혼선이 줄어들었다.
- 프로젝트를 진행하는 데 큰 도움이 되었다.



감사합니다.



데이터 관리 (FE)

- redux → context
 - compile error
 - context 만으로 충분히 구현 가능
- 3개의 context 사용

panel	<ul style="list-style-type: none">● navigate● 패널 데이터
auth	<ul style="list-style-type: none">● jwt 토큰 관리
user	<ul style="list-style-type: none">● 사용자 아이디● 프로필 목록● 현재 프로필● 선호 식당 목록● 방문 식당 목록



유저와 프로필 분리

하나의 유저에 여러 개의 프로필이 가능해야 한다는 요구사항

- 유저와 프로필을 구분하여 DB에서 관리, 개별적인 jwt 토큰 발급 및 관리

	유저 토큰	프로필 토큰
발급 시기	로그인 시	프로필 선택 / 전환 시
토큰 포함 요청	프로필 관리 요청 (생성, 변경, 삭제)	사용자 정보 필요 요청 (좋아요, 방문목록 등)



Backend Framework & DB

제한된 시간과 적은 인원 / 변경의 여지가 많은 데이터 구조

- Express.js: 간단하고 직관적인 구조로 빠른 개발 가능
- MongoDB: 유연한 schema로 빠르고 편한 데이터 구조 변경 가능