

안녕하십니까!

진심을 사랑하는 개발자, 한준범입니다.



세상 곳곳에 숨겨져 있는 균열을 메꿔주기 위해 소프트웨어 개발에 발을 들이게 되었습니다.

많은 프로젝트의 **서비스 기획 / 개발 / 배포 / 운영** 경험이 있으며, 여러 박람회에서 열변을 토하며 발표한 경험이 있습니다. 모두의 성장을 위해서라면, **상호 존중**이 기반이 되어야 한다고 생각합니다. 그렇기에 어떤 행동을 하기에 앞서 배려 / 존중 / 공감을 선행하는 것을 사랑하며 무엇이든지 진심이 있게 행동하는 것 또한 매우 사랑합니다.

기술

프로그래밍 언어

golang typescript java python c

API 및 통신

HTTP Websocket socket.io webrtc grpc

데이터베이스

mysql maria postgresql cassandra bolt

프레임워크

nestjs spring boot gin echo fiber

협업 도구

git github gitflow slack figma

학력

대구소프트웨어마이스터고등학교

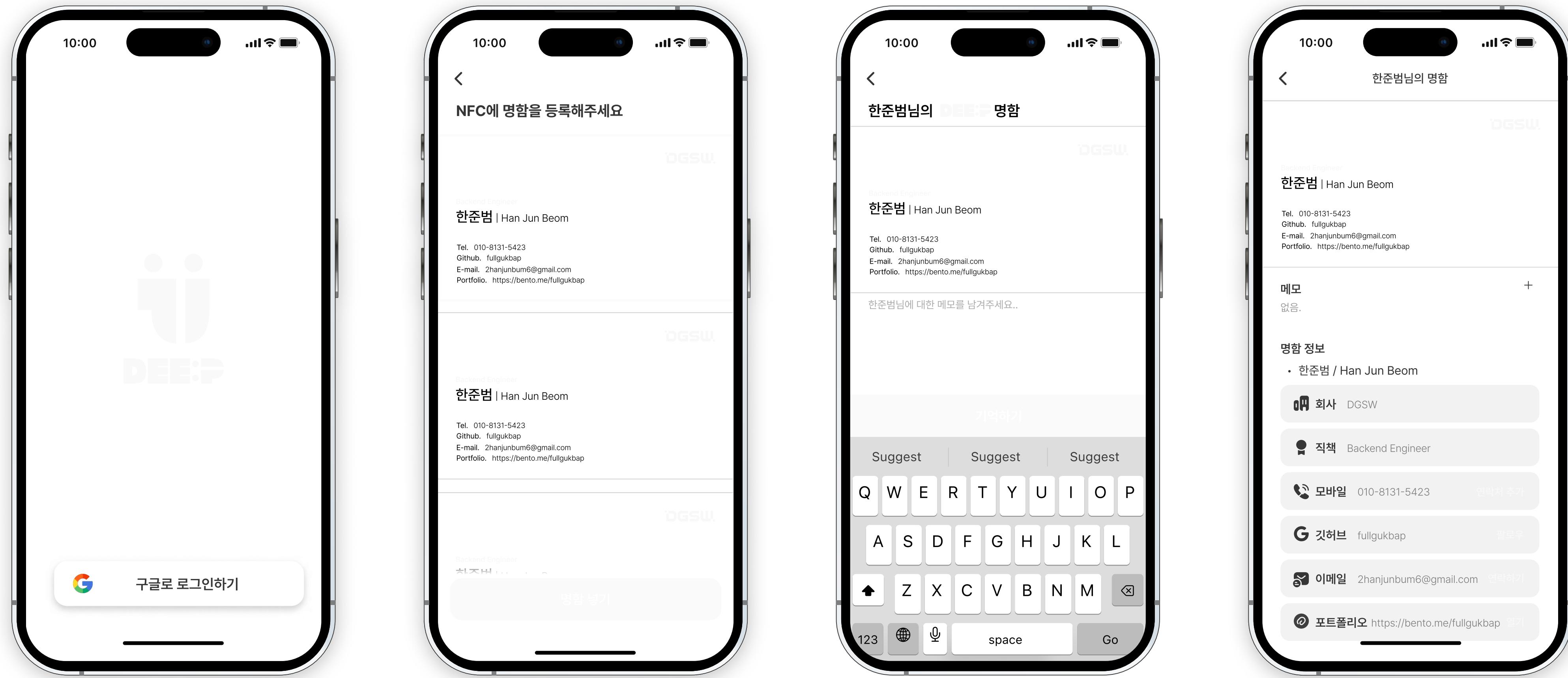
2022. 03 ~ 2025. 02

소프트웨어개발과

- 자료구조, 알고리즘, 데이터베이스, 네트워크, 컴퓨터구조 등 기초 지식 습득
- 교내 교육봉사 동아리 두카미에서 중학교 학생 대상으로 봉사 활동 총 7번
- 카카오, 카카오워크, 더스윙 등 다양한 기업에 방문

DEE:P

종이명함의 한계점을 NFC로 무한하게



🚀 개발 배경

문제인지

소프트웨어마이스터고등학교는 국내에 총 4개의 학교가 있으며, 이 학교들은 매년 모든 학생이 한자리에 모여 서로를 알아가는 교류 행사를 진행합니다. 그 중 한 프로그램으로 종이 명함 교환 시간을 있었습니다. 이때 몇몇 학생들이 예상치 못한 불편함을 겪는 모습을 볼 수 있었습니다.

- 명함을 실수로 떨어트려 교환에 집중하지 못하고 좁고 있는 학생
- 준비한 명함 개수가 적어 교환을 더 이상 할 수 없는 학생
- 명함이 물에 젖거나 찢어져 사용하지 못하는 상황에 처한 학생

이처럼 종이 명함은 물리적 한계와 환경적 한계의 문제점이 있다고 인지했습니다.

해결방안

종이 명함의 한계점을 해결하기 위해 저희 팀원들은 방안에 대해 토의하기 시작했습니다. 그 과정에서 아이디어들이 폭포처럼 쏟아졌고 그 결과 “**NFC를 이용해 명함을 공유하는 건 어때?**”라는 기발한 생각을 해냈습니다. 그래서 저희는 **NFC를 이용해 위의 한계점을 해결할 수 있음을 확신했고** 더 나아가 무한대에 가까운 명함 공유 서비스를 만들 수 있다고 생각했습니다.

다른 서비스와 경쟁력

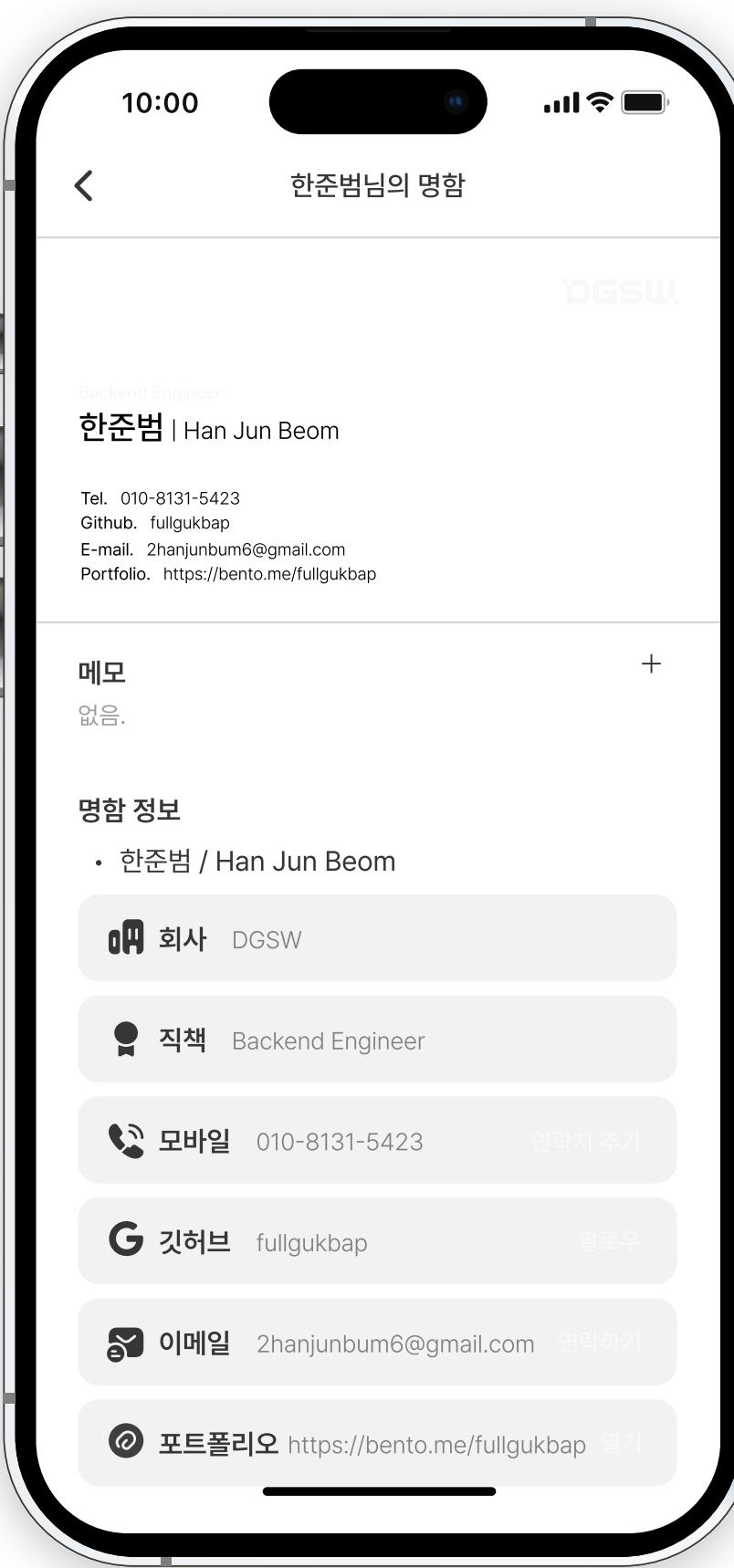
그러나 명함을 소재로 사업하는 기업들은 매우 많았습니다. 그중 특히 **슬라이스라는 기업이 가장 눈에 들어왔습니다**. 이 기업은 저희와 같은 아이디어로 이미 서비스하고 있었기 때문입니다. 이에 저희는 어떻게 차별화할지 고민한 끝에 스티커를 활용하기로 했습니다. 슬라이스는 트렌디한 이미지를 내세우며 NFC 명함을 약 30,000원의 고가에 판매하고 있었지만, 저희는 저렴한 옵션을 원하는 소비자층을 공략하기로 했습니다. 또한 스티커는 지갑이나 휴대전화에 쉽게 붙일 수 있어 용이합니다.

대표적인 기능들



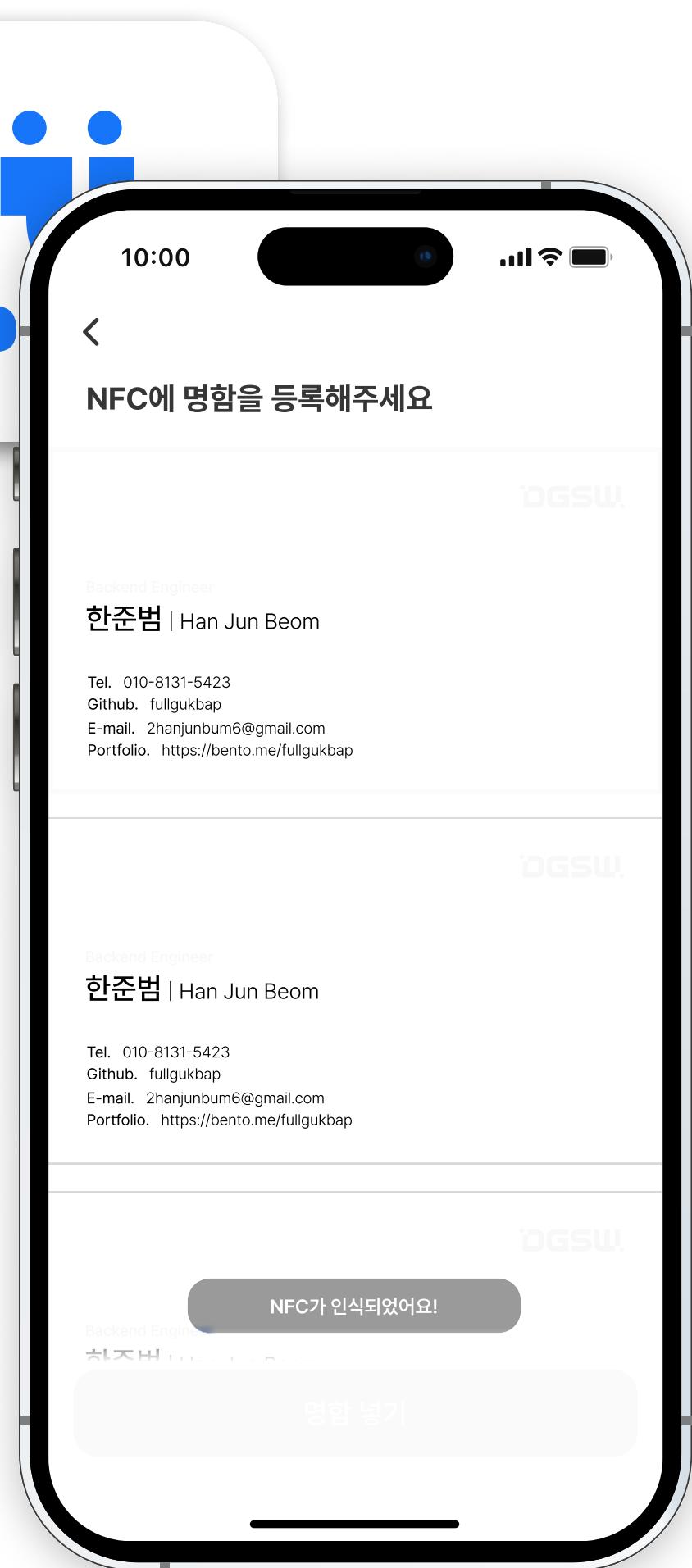
로그인 / 회원가입

Oauth를 이용해 사용자들이
이 간편하게 로그인 및 회원
가입을 할 수 있도록 UX를
설계했습니다.



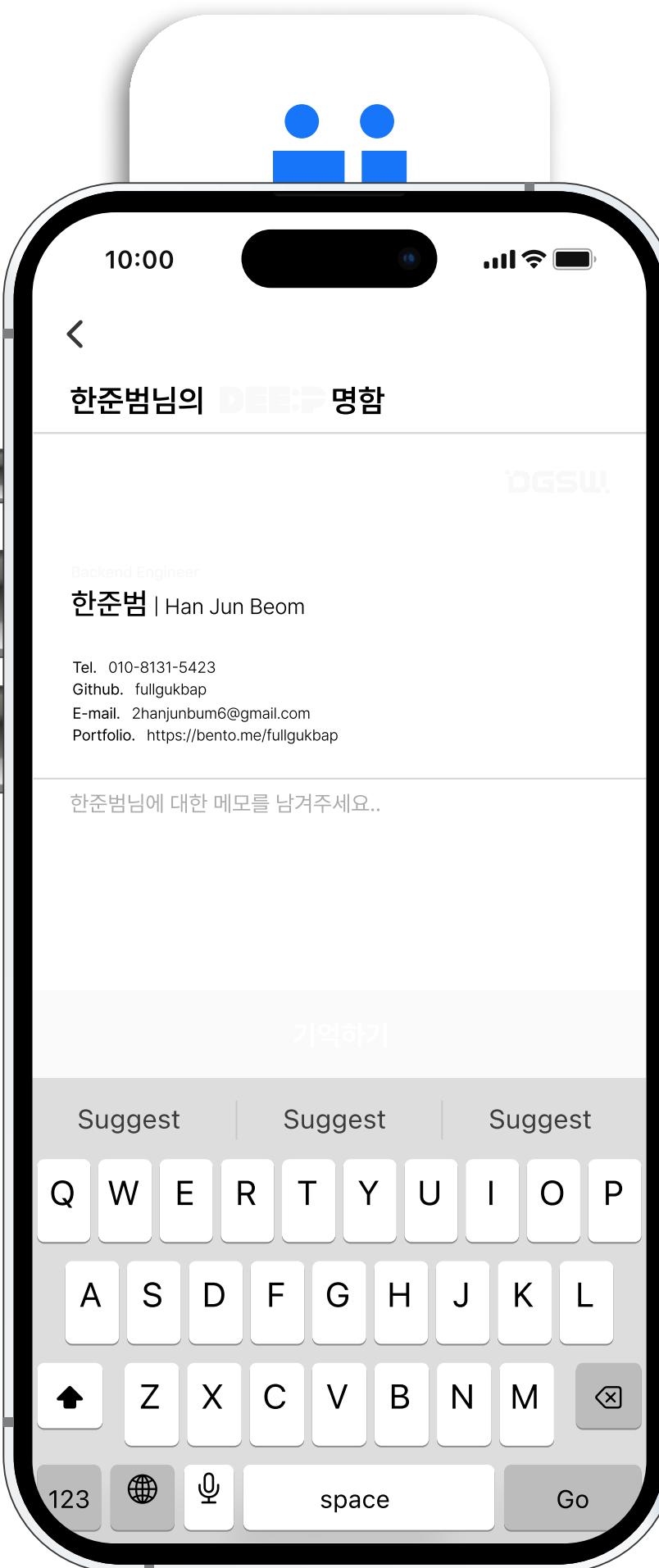
명함 보기

사용자는 교환한 명함들을
리스트로 확인할 수 있으며,
특정 명함의 세부 정보를 한
눈에 확인할 수 있습니다.



스티커에 명함 삽입하기

사용자는 구매한 스티커에
자신의 명함을 삽입할 수 있
습니다. 삽입 방법은 두 가지
로, DEEP 서비스에서 제공
하는 템플릿을 사용해 명함
을 제작하거나, 휴대폰으로
직접 촬영해 명함 사본을 만
들 수 제작할 수 있습니다.
사본을 제작할 경우 Naver
Cloud의 OCR 기술을 활용
해 정보를 카테고리별로 추
출하고 저장합니다.



DEEP 스티커 인식하기

다른 사용자의 DEEP 스티
커를 인식하면, 해당 스티커
소유자의 명함 정보가 표현
됩니다.

The screenshot shows a web-based template creation interface for business cards. It includes fields for Name, Position, Department, Phone Number, Email, and GitHub/Homepage. A callout bubble from the template area points to a message: "사용자들은 웹에서 템플릿을 이용해 손쉽게 제작할 수 있어요" (Users can easily create it using the template on the web). The right side of the interface shows a preview of the generated business card for "한준범 | Han Jun Beom" with contact details.

한준범님의
DEEP 명함 제작하기

이름 *
ex) 홍길동

직책 *
ex) 홍길동

부서 *
ex) 홍길동

전화번호 *
ex) 홍길동

이메일 *
ex) 홍길동

GitHub / Homepage
ex) 홍길동

깃허브를 입력해주세요

생성하기

한준범
사용자들은 웹에서 템플릿을 이용해 손쉽게 제작할 수 있어요

명함 미리보기

한준범 | Han Jun Beom

Tel. 010-8131-5423
Github. fullgukbap
E-mail. 2hanjunbum6@gmail.com
Portfolio. https://bento.me/fullgukbap

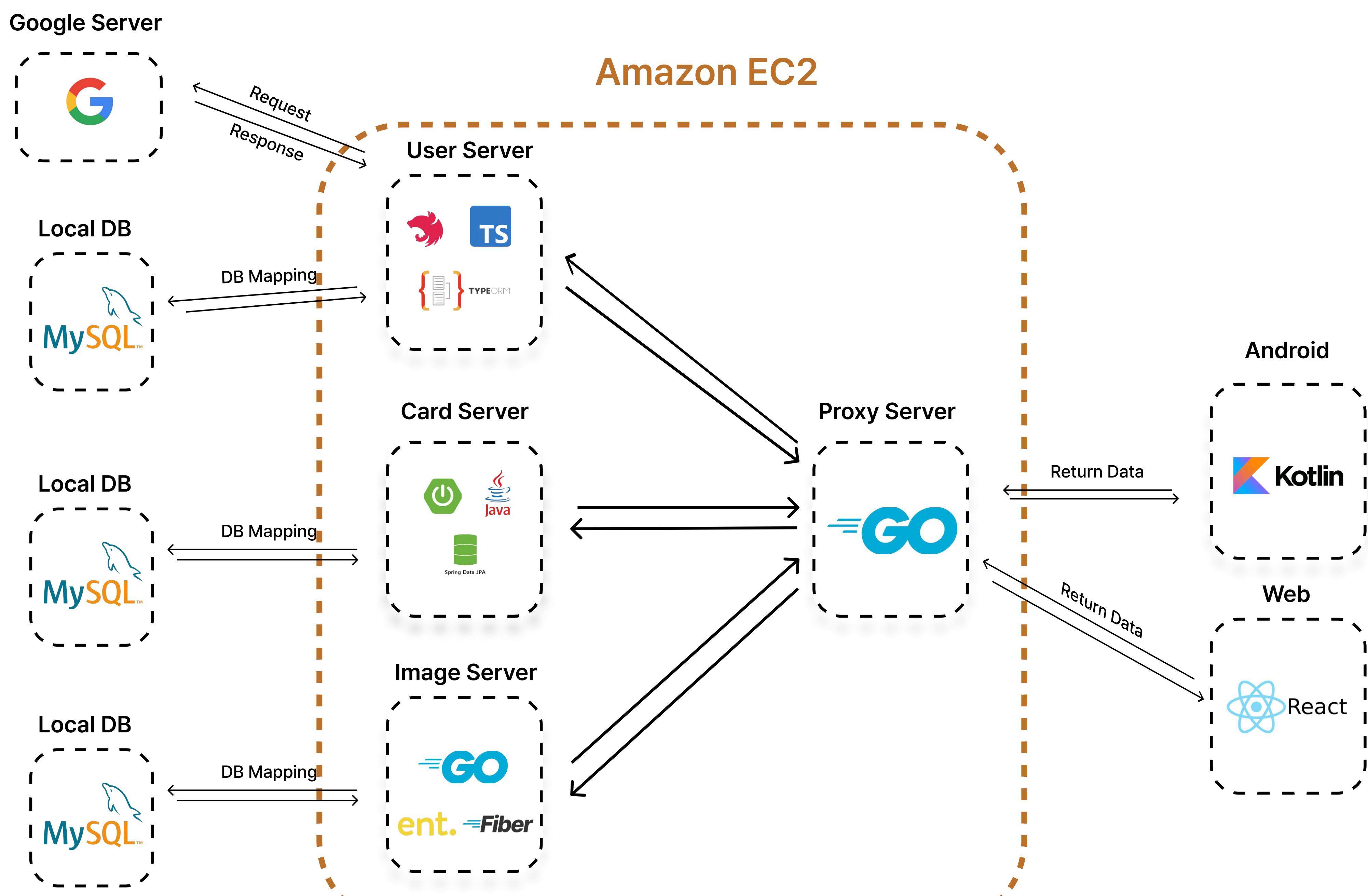
🔨 개발 팀원 구성원

Backend 3명(본인포함) Frontend 2명 Android 2명

🔨 사용한 기술

- golang java typescript
- fiber http/net spring boot nestJS
- typescript react touter recoil styled-components
- mysql ent jpa typeorm
- Naver CLOVA OCR (이미지 속 문자 추출) aws tmux

🚧 시스템 아키텍처



🏗 주요 개발 및 기여

클라이언트의 요청을 적절한 마이크로서비스에 전달하여 처리할 수 있도록 하는 MSA의 스(Proxy Server)를 개발했습니다.

명함의 이미지를 저장하고 조회하고 수정하고 삭제하기 위하여 직접 이미지 서버를 개발했습니다.

- PNG 형식의 이미지(크기: 3.5KB)를 요청하고 로드하는 데 총 4.12ms가 걸렸습니다. (리소스 예약: 1.18ms, 연결 시작: 0.99ms, 요청 및 응답: 2.51ms)

AWS EC2를 이용해 각각의 마이크로 서비스의 배포의 역할도 맡았으며, tmux를 사용하여 체계적이고 관리적인 환경을 구축했습니다.

ICT 박람회 및 Softwave에 참여하여 박람객분들께 소개하며, 발표 세션에 참여하여 발표한 바 있습니다.

⭐ 문제 및 해결책

일일이 API 등록하려니깐 너무 불편한데...

프록시 서버(Proxy Server)를 개발할 당시 일일이 마이크로 서비스들의 API들을 직접 등록하여 작동시키게끔 설계했습니다. 이러한 방법을 통해 배포하게 되니 마이크로 서비스들 중의 하나라도 API가 변경되었을 때 다시 코드를 작성해 배포에 시간에 많은 시간을 한다는 문제점이 보였습니다.

그래서 본 문제를 해결하기 위해 파일 기반의 라우팅 관리 방식을 도입했습니다. 각 마이크로서비스에 대한 URL 매핑 정보를 개별 파일로 저장하여 라우팅 설정의 추가와 관리를 더욱 용이하게 만들었습니다. 이를 통해 시스템의 유연성과 유지 보수성이 크게 향상되었으며, 배포 시간 또한 크게 단축되었습니다.

```
// 일일이 등록 과정을 거쳐야 했음
http.HandleFunc("POST /v1/api/auth/id-check", func(w http.ResponseWriter, r *http.RequestOnBehalf{w, r, "POST", "http://127.0.0.1:8081/v1/api/auth/id-check"})
{
    RequestOnBehalf{w, r, "POST", "http://127.0.0.1:8081/v1/api/auth/signup"}
}

http.HandleFunc("POST /v1/api/auth/validate", func(w http.ResponseWriter, r *http.RequestOnBehalf{w, r, "POST", "http://127.0.0.1:8081/v1/api/auth/validate"})
{
    RequestOnBehalf{w, r, "POST", "http://127.0.0.1:8081/v1/api/auth/validate"}
}
```

```
{
    "target": {
        "name": "user",
        "ip": "http://127.0.0.1",
        "port": ":8081"
    },
    "endpoints": [
        {
            "method": "POST",
            "pattern": "/v1/api/auth/id-check"
        },
        {
            "method": "POST",
            "pattern": "/v1/api/auth/signup"
        },
        {
            "method": "POST",
            "pattern": "/v1/api/auth/validate"
        }
    ]
}
```



파일로 등록할 수 있게 코드 변경

AWS 비용이 예상했던 가격보다 왜 이렇게 많이 나왔지?

DEEP 서비스는 AWS의 EC2 서비스를 통해 배포되었습니다. 그러나 예상했던 비용보다 훨씬 높은 청구 금액을 보고 깜짝 놀랐습니다. 처음에는 명함 교환으로 때문에 트래픽이 급증한 것으로 생각했지만, 직접 분석해보니 원인은 달랐습니다. 트래픽 증가가 아닌 여러 리전에서 EC2 서버를 실행하면서 비용이 많이 늘어난 것이었습니다. 문제를 해결하기 위해 AWS에 상황을 설명하고, 실수였음을 솔직히 전한 결과, AWS 측에서 이를 이해해주고 환불해주셨습니다.

🚢 느낀점

처음으로 MSA를 설계 및 개발을 해보면서 서비스 간의 독립성의 효용가치가 높고, 유연성과 확장성에 매우 적합한 아키텍처임을 깨달았습니다.

Go 언어를 사용해 유ти리티 서버를 빠르게 개발하고, 구상한 아이디어를 손쉽게 구현할 수 있어서 매우 강력한 언어임을 느꼈습니다.

서비스를 시장에 출시하는 과정은 매우 어렵지만, 주기적인 사람들과의 만남과 다양한 경험을 통해 지속적으로 고객의 니즈를 파악해야만 서비스의 가치를 더욱 증대시킬 수 있다는 것을 깨달았습니다.

어떠한 문제점이 발생했을 때, 단순히 넘기는 것이 아니라 그 문제의 본질을 파악하고 해결 방안을 모색하며, 팀원들과 함께 아이디어를 구현하는 이 과정이야말로 프로그래머로서 해야 할 일임을 깨달았습니다. 팀원들과 함께 프로덕트를 하루하루 만들어가면서, 세상의 다양한 문제를 해결하고 상처를 치유하는 이 과정에 깊이 매료되었습니다.

프로젝트 소스 코드

organizations github.com/orgs/Project-DEE-P/
proxy-server github.com/Project-DEE-P/DEEP-backend-proxy
image-server github.com/Project-DEE-P/DEEP-backend-image

TOKTOK

세상에 외로운 사람을 위해 메타버스 속에서 새로운 만남을



🚀 개요

거주 지역은 충청북도이고, 학교는 경상북도입니다. 방학 때는 친한 친구들과 만날 수 없어 매우 **외로운 시간**을 보내야 했습니다. 이러한 문제를 해결하기 위해 **메타버스 소재**를 이용해 사람들과 손쉽게 교류할 수 있는 프로덕트를 제작하려 했지만, **설계상의 문제**로 때문에 결국 개발을 완전히 종료하게 되었습니다.

🔨 사용한 기술

- golang fiber ent websocket
- testing testcontainer validator
- raspberrypi

🍼 개발 팀원 구성원

프론트 엔드 개발자 1명, [백엔드 개발자 1명 \(본인\)](#)

🏗️ 주요 개발 및 기여

TOKTOK 프로젝트를 진행하면서 **팀장의 역할을 맡았습니다**. TOKTOK을 개발하기 위해 해내야 할 일을 정리하였고, 이것을 기반으로 회의를 진행하며 **팀원들을 이끌었습니다**.

Hexagonal Architecture를 도입하여 테스트의 용이성을 증폭시키고, 다른 개발자가 참여할 시 코드를 빠르게 이해하고 바로 프로젝트에 참여할 수 있게 했습니다.

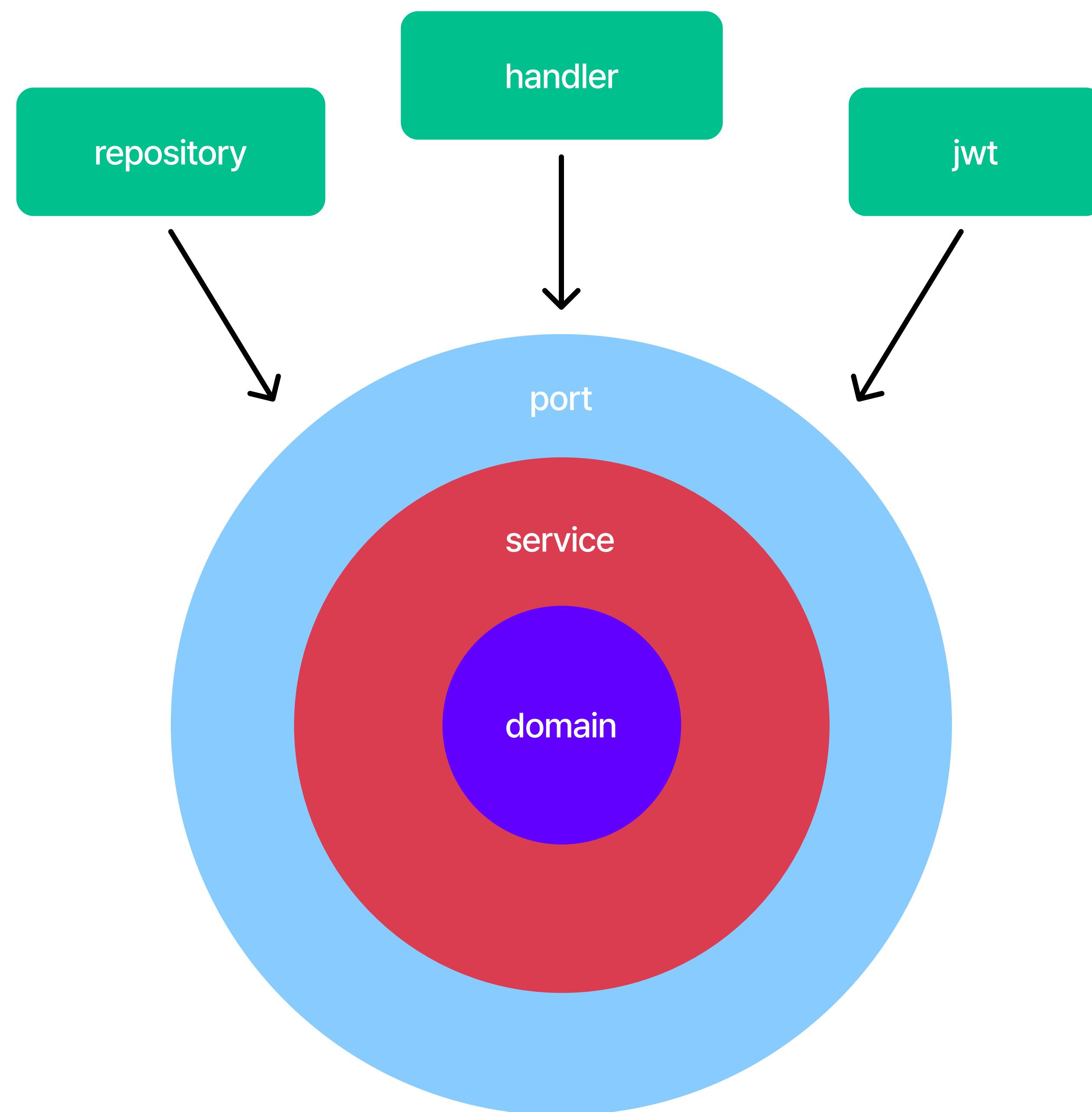
testcontainer-go를 이용해 **Repository Layer**를 테스트한 바 있습니다.

실지적인 서비스를 구현하기 위해 중복 로그인 방지, 데이터베이스 저장 시 암호화하여 저장, 서비스를 사용하기 위해 동의해야 하는 약관 기술, App Push들을 구현하기 위해 팀원 정보를 공유하여 구현 가능성을 높였습니다.

🏗️ 소프트웨어 구조 - 서버

앞서 언급했듯이 TOKTOK은 Hexagonal Architecture를 도입해 개발을 진행했습니다.

아래의 도식화는 설계도입니다.



위와 같은 설계를 아래와 같은 구조로 개발했습니다.

```
core
  domain
    domain_avatar.go
    domain_message.go
    domain_relation.go
    domain_room.go
    domain_user.go
    errors.go
    token_payload.go
  port
    port_auth.go
    port_avatar.go
    port_message.go
    port_relation.go
    port_room.go
    port_user.go
  service
    service_room.go
    service_auth.go
    service_avatar.go
    service_realtion.go
    service_user.go
  utils
    encryption.go
```

```
persistence
  mysql
    client.go
    migration.go
  repository
    repository_avatar.go
    repository_message.go
    repository_relation.go
    repository_room.go
    repository_user.go
  test
    provider
      provider.go
      provider_test.go
    repository_avatar_test.go
    repository_relation_test.go
    repository_user_test.go
    repository_utils.go
  utils
    convert.go
    error.go
    utils.go
```

repository layer(adapter)를 구현한 모습

```
handler
  fiber
    catcher
      custom_error_handler.go
      error_response.go
      error_set.go
    controller
      controller_auth.go
      controller_avatar.go
      controller_chat.go
      controller_relation.go
      controller_room.go
      controller_user.go
    dto
      auth_request.go
      auth_response.go
      avatar_request.go
      avatar_response.go
      relation_request.go
      relation_response.go
      room_response.go
      user_request.go
      user_response.go
    middleware
      middleware_jwt_validation.go
    realtime
      chat
        chat_system.go
        client.go
        request.go
        room.go
        send_message.go
    router
      router.go
      router_controllers.go
      router_services.go
    validator
      validator.go
```

handler layer(adapter)를 구현한 모습

domain, port, service를 정의한 모습

아키텍처를 사용해본 느낀점

집요하게 Hexagonal Architecture 조건을 따를 필요가 없다고 느꼈습니다. 본 아키텍처는 복잡성을 줄이기 위한 목표이지, 원칙대로 하다 보면 더 복잡해질 때도 있다는 것을 느꼈으며, 시간과 돈이 투자되기 때문에 선택과 집중이 필요하기 때문입니다.

테스트 코드의 힘을 깨달았습니다. Hexagonal Architecture의 테스트 용이성을 이용하기 위해 Repository Layer를 테스트했습니다. 테스트를 통해 생각하지 못한 경우의 수에 대해 미리 대책을 세울 수 있었고, 그 결과 코드에 정확도가 높아져 더 자신감있고 빠른 개발을 진행할 수 있었습니다.

Hexagonal Architecture는 협업을 용이하게 만듭니다. 도메인과 포트를 정의하는 단계에서 인터페이스를 명확히 설정하면, 팀원들은 각자의 역할에 집중할 수 있고, 포트를 통해 의존성을 관리할 수 있기 때문입니다.

개발을 종료하게 된 계기

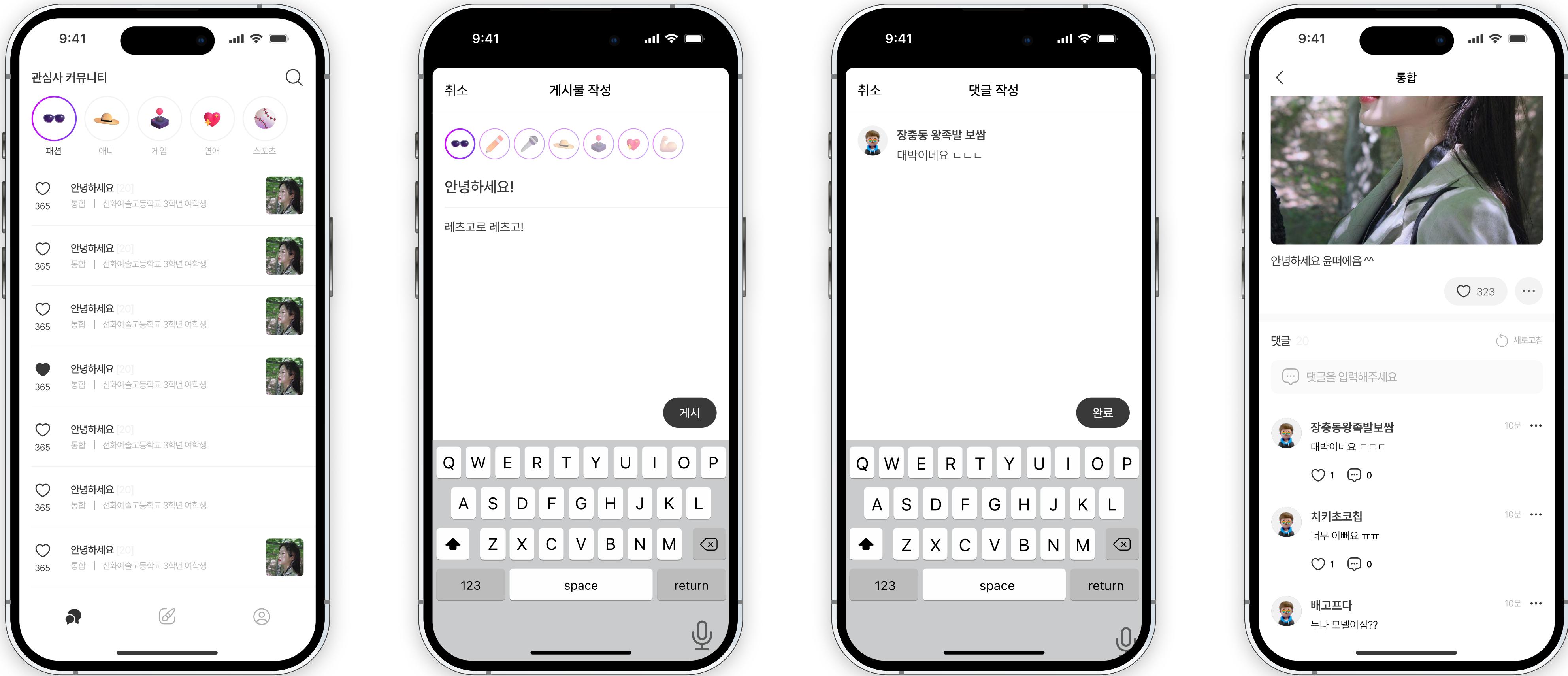
TOKTOK 프로젝트는 외로움으로부터 시작되었습니다. 그러나 외로움을 해결하기 위해서는 메타버스도 도움을 줄 수는 있다고 생각하지만, **해결책은 아니라고 결론을 짓게 되었습니다.** 그리하여 TOKTOK 개발을 종료하게 되었고, 이 과정을 통해 문제 정의, 문제를 해결하는 아이디어가 시장에서 유효한지 미리 검증하는 과정도 매우 중요하다는 것을 깨달았습니다.

프로젝트 소스 코드

organizations github.com/orgs/he-he-heng/repositories
toktok-backend github.com/he-he-heng/toktok-backend

Letsgo

오직 학생들만을 위한 커뮤니티 서비스



🚀 개요

학생들은 대부분 인스타그램, 페이스북과 같은 커뮤니티 서비스를 이용하고는 합니다. 그러나 오직 학생들은 위한 소셜네트워크 서비스는 존재하지 않았습니다. 그래서 저희는 학생과 학생들의 커뮤니티, 더 나아가 학교와 학교들이 서로 상호작용하여 새로운 가치를 창출해내는 서비스를 만들고 싶어 Letsgo라는 프로젝트를 진행하게 되었습니다.

- 사용자들은 관심사 커뮤니티에 자신의 최근 소식을 확인 및 게시할 수 있다.
- 사용자들은 자신의 학교 커뮤니티 안에서 자신의 최근 소식을 확인 및 게시할 수 있다.
- 학교와 학교가 대결하는 학교 대항전 이벤트를 통해 학교의 응집력을 확인할 수 있다.

🔨 사용한 기술

- React Typescript Stable-Diffusion
- Redux Toolkit Encrypted Storage Styled Component
- NestJS Prisma mysql Naver Cloud Platform

🍼 개발 팀원 구성원

프론트 엔드 개발자 4명, 백엔드 개발자 3명 (본인포함)

🏗️ 주요 개발 및 기여

처음으로 접해보는 nestjs와 prisma를 공시문서를 이용해 빨리 익히고, 모르는 게 있으면 물어보는 방법을 통해 데드라인에 맞춰 개발을 진행했습니다.

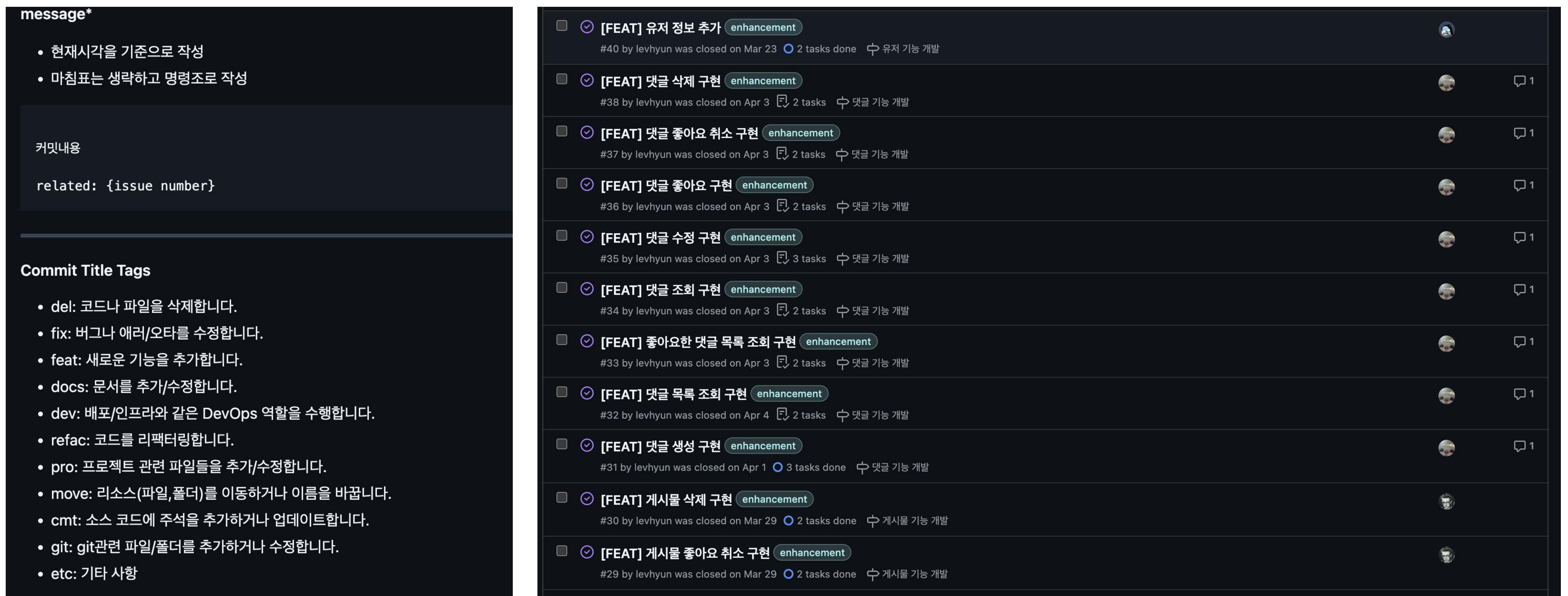
gitflow를 이용해 자기의 해야 할 일을 issue로 관리하여 협업에 용이하게 하였습니다.

프로젝트의 배포 역할을 맡았으며, 도커화와 Naver Cloud Platform에 개발한 서버를 배포하였습니다.

AI Seoul Expo에 참여하여 개발한 프로젝트를 전시하여 발표한 바 있습니다.

Git-Flow 협업 방식

저희 팀원들은 **branch** 관리와 원활한 협업을 하기 위해 Git-Flow를 도입하였습니다. 또한, 전체적인 일관성을 추구하기 위해 **Git-Convention**을 지정하는 과정도 거쳤습니다.



The screenshot shows a GitHub pull request interface. On the left, there's a 'message*' field containing a bulleted list of conventions: '현재시각을 기준으로 작성', '마침표는 생략하고 명령조로 작성'. Below it is a '커밋내용' section with placeholder text 'related: {issue number}'. To the right is a detailed list of commits, each with a purple circular icon and the text '[FEAT]'. The commits are categorized under 'enhancement' and include descriptions like '유저 정보 추가', '댓글 삭제 구현', '댓글 좋아요 취소 구현', etc. The commits are dated from March 23 to April 3, with various numbers of tasks assigned. At the bottom of the commit list, there's a 'Commit Title Tags' section with a long list of abbreviations and their meanings, such as 'del', 'fix', 'feat', 'docs', 'dev', 'refac', 'pro', 'move', 'cmt', 'git', and 'etc'.

느낀 점

처음으로 접한 프레임워크 및 도구들을 익히는 것은 매우 어렵지만, 팀원들이 덕분에 손쉽게 해결해나갈 수 있었습니다. 그렇기에 **팀원은 매우 소중한 존재임**을 다시 한번 느끼게 되었습니다.

gitflow를 적용해 개발하니 정말 체계적으로 개발할 수 있었고 무엇보다 **코드리뷰하기가 매우 편리했습니다.**

코드리뷰를 통해 팀원들이 어떻게 작성했는지 보고, 피드백을 즉각 주고받을 수 있어서 매우 편리하였고, **서버의 질을 올리는데 있어서 큰 도움이 되었습니다.**

NestJS와 Prisma를 사용해본 결과 매우 생산적인 프레임워크라는 것을 깨달았습니다.

프로젝트 소스 코드

[letsgo-api](#)
[organization](#)

github.com/TW00D/letsgo-api-mini
github.com/orgs/TW00D/repositories

수상	2022 교내 알고리즘 대회 CAC 은상 2022 국립대구청소년디딤센터 자원봉사상 2022 제4회 SW융합 학생 해커톤 융합상 2023 1학기 교내 해커톤 금상 2023 교내 우수 ICT 프로젝트 2등상 2023 교내 우수 ICT 프로젝트 3등상 2학년 교내 다독상 (1, 2 학기) 대구교통공사 DTRO 장학생 선정	2022. 02. 09 2022. 07. 04 2022. 10. 21 2023. 02. 07 2023. 11. 09 2023. 11. 09 2023. 10. 29 2023. 10. 31
활동	국립대구청소년디딤센터 봉사활동 Smart App Challenge 2022 참가 제과제빵 봉사활동 참여 2022학년도 1학년 노벨 엔지니어링 해커톤 2022 대소마고 Software 축제 해커톤 2023 교내 우수 ICT 프로젝트 3등상 2022 제4회 SW 융합 학생 해커톤 두카미 레퍼런스 진행 카카오 스페이스 방문 KAKAOINTERPRISE 회사 견학 데이터베이스 ERD설계 프로젝트 SFPC 알고리즘 대회 모디 C언어 수업 진행 Hackers ground 해커톤 참여 2023 융합 노벨 엔지니어링 해커톤 I/O Extended 2023 Seoul 참여 2023 IMPACT_CONNECT 참여 GopherCon Korea 2023 마이스터 4개교 연합 토크콘서트 2023 ICT 융합 엑스포 부스 운영 ICT 박람회 참석 및 부스 운영 swing 기업탐방 교내 B1CODE 컨퍼런스 대회 세션 발표 2023 대한민국 소프트웨이브대전 부스 운영 2024 국제인공지능대전 부스	2022. 04. 12 ~ 2022. 07. 12 2022. 05. 12 ~ 2022. 05. 15 2022. 05. 14 2022. 07. 17 2022. 07. 19 ~ 2022. 07. 20 2023. 11. 09 2023. 10. 29 2022. 11. 03 2022. 11. 14 2022. 11. 18 2022. 11. 21 ~ 2022. 12. 05 2023. 01. 12 ~ 2023. 01. 13 2023. 03. 06 ~ 2023. 07. 21 2023. 06. 21 ~ 2023. 06. 23 2023. 07. 12 ~ 2023. 07. 13 2023. 07. 29 2023. 08. 04 2023. 08. 05 2023. 08. 24 2023. 10. 23 ~ 2023. 10. 26 2023. 11. 08 ~ 2023. 11. 10 2023. 12. 01 2023. 12. 24 2023. 11. 19 ~ 2022. 12. 01 2024. 05. 14 ~ 2024. 05. 16
자격증	정보처리산업기사 운정면허 보통 1종	2024. 05. 03 2024. 08. 02