(같이 들어가서 인사하기!) 안녕하세요! SurFinger 입니다.!
안녕하세요 SurFinger팀의 발표를 맡은 정성민입니다.

목차

1 팀원 구성 및 역할분담

2 프로젝트 목표

3 슬랙 봇 설명

4 SurFinger 데모

5 구현된 내용

6 성과 및 향후 진행 방향

7 QnA

네 다음은 저희 발표의 목차이구요. 프로젝트 목표에 대한 설명으로 시작해서 저희가 선택한 슬랙에서의 봇 구조를 간단히 말씀드리고 개발된 SurFinger를 시연한 다음 구현된 내용과 성과를 말씀드리고 마치겠습니다.

팀 구성

최평강
팀원

정성민
팀장

이태우
팀원

저희 SurFinger 팀의 구성입니다. 제가 팀장을 맡았고 평강이형 태우형과 함께 프로젝트를 진행하였습니다.

프로젝트 목표

네! 이제 프로젝트 목표에 대해 설명드리도록 하겠습니다.

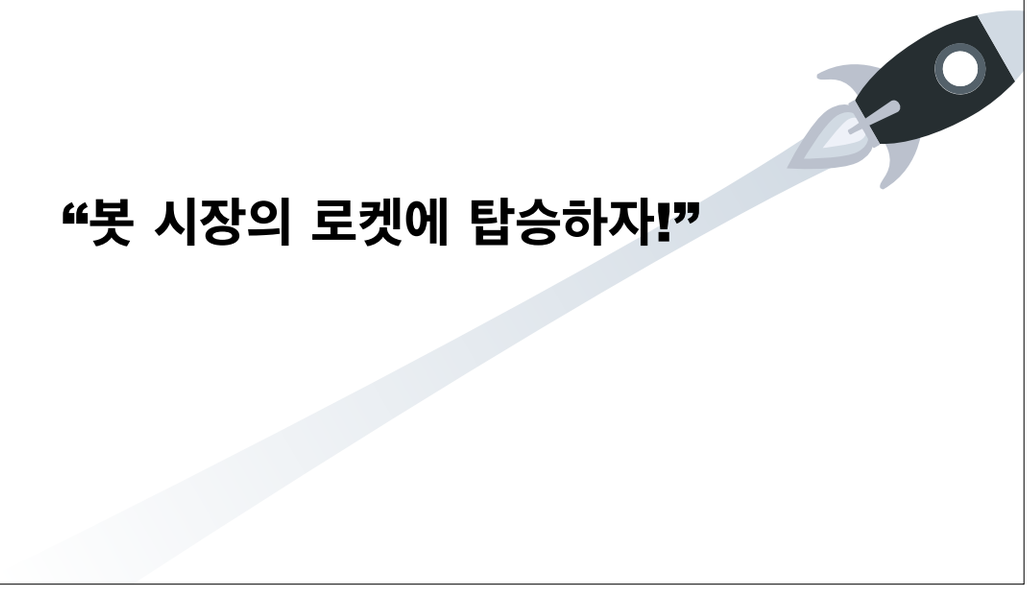
봇 시장 소개

"끊임 없이 성장하는 챗봇 시장"

(출처:'chatbot Google Trend')

구글 트렌드에 따르면 챗봇에 대한 관심도는 점점 증가하고있으며, 2013년에 비해 관심도가 3배 증가한 것을 볼 수 있습니다. 뿐만아니라 페이스북 구글 마이크로소프트 등 여러 기업들도 챗봇에 투자를 하며 다양한 챗봇들이 나오고 있습니다.

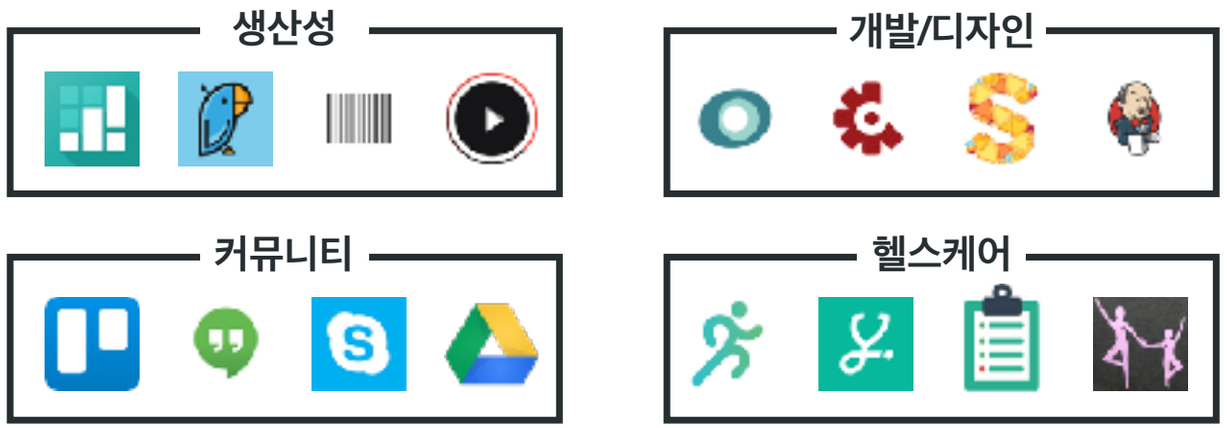
프로젝트 목표



“봇 시장의 로켓에 탑승하자!”

이러한 성장 추세에 힘 입어 저희 팀도 봇 시장의 로켓에 탑승 해보자라는 목표를 갖고 프로젝트를 진행해보게 되었습니다.

무슨 봇들이 있을까?



“그런데…”

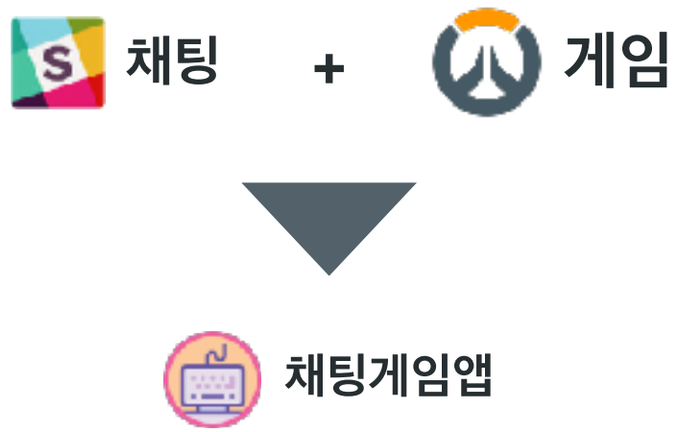
그래서 어떤 봇들이 있을까 조사를 해 보았습니다. 생산성 마케팅 디자인 커뮤니케이션 헬스케어 등 여러 분야의 봇들이 존재하고있었습니다. 그런데….

게임봇 이 없어!

개발의 꽃인 게임이없다고!

게임 봇이 없었습니다! 개발의 모든 분야의 집약체인, 개발의 꽃이라 불리는 게임이 없었습니다. 아! 그럼 우리는 게임봇을 만들어 보면 되겠구나? 라고 생각했습니다.

어떤앱을 만들까?



그래서 어떤게임을 만들까? 채팅과 게임을 합해 어떻게 구현을 해야할까?

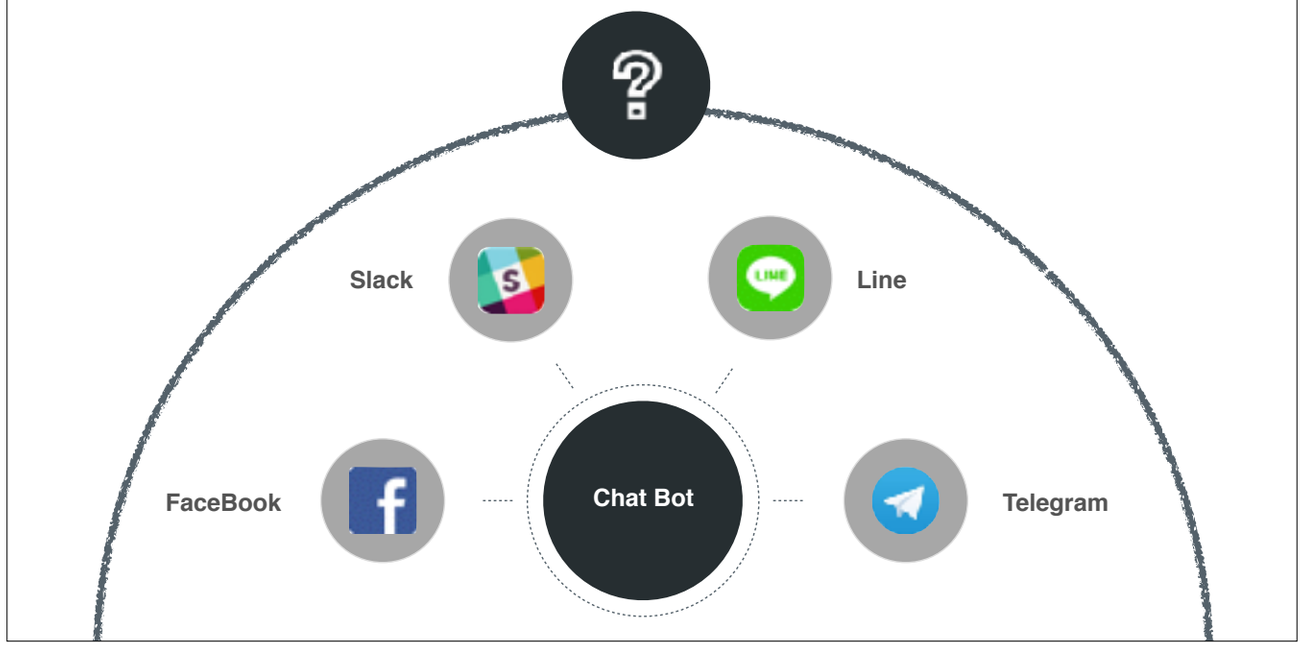
채팅은 텍스트 기반으로 하는건데 텍스트게임이라면, 타자를 쳐서 맞추는 타자게임 어떨까? 라고 생각해 채팅게임을 만들어보게되었습니다.

채팅 게임앱의 역사는..

Dos **win 95~2000** **window xp~** ?

생각해보니 타자게임은 도스시절부터 많이 즐겨왔고, CUI에서 GUI로 넘어온 뒤에도 많이 사랑받아왔습니다. 저희는 분명 다음세대의 타자게임은 챗봇형태일 것이라고 생각합니다.

많은 플랫폼들.. 그중에서?



그래서 채팅게임을 만들기로 결정을 했는데, 세상에는 수 많은 채팅 플랫폼들이 존재합니다. 그 수많은 플랫폼 중에서 저희는 슬랙을 하기로 결정 하였습니다. 그 이유는 ….

슬랙을 선택한 이유

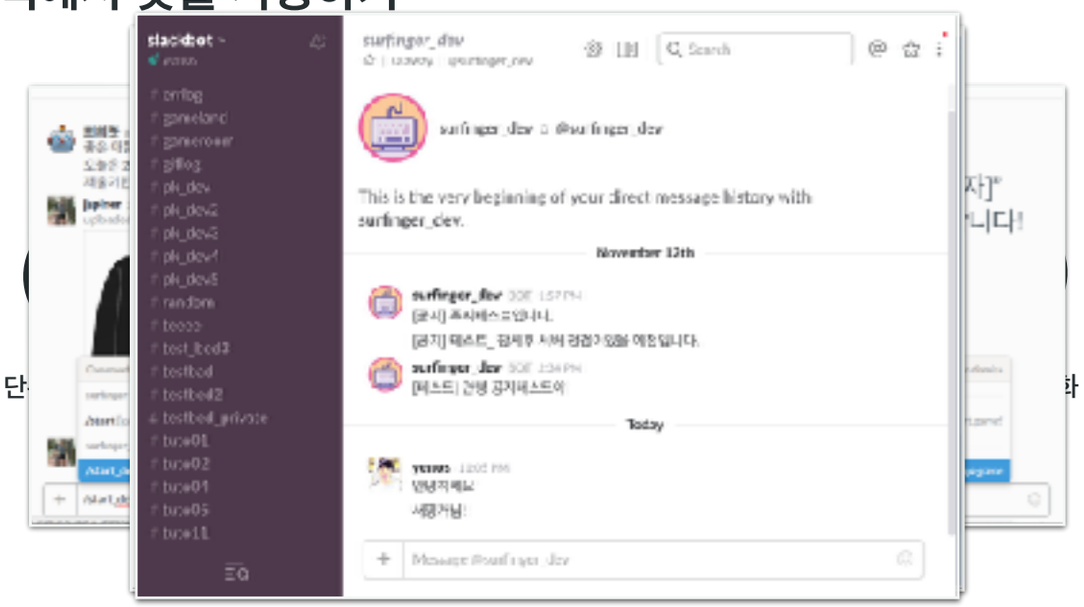
Webhooks	Web API	Messaging API
Send API	Real Time Messaging API	Webhook API
Web Views	Group based chatting	Push API
Plugins	User Interaction	Reply API
Thread Settings	Events API	
User Profile	Methods	
	SCIM API	
<Facebook BOT>	<Slack BOT>	<LINE BOT>

채팅 게임을 만들기 위해선 '실시간성'과 '그룹' 단위의 플레이가 필요했습니다. 슬랙에서는 real-time api 외에도 다양한 api를 지원할 뿐만 아니라 플랫폼 자체가 그룹 단위의 시스템이었기에 게임을 만들기에 최적이라 판단해 슬랙을 선택하였습니다.

슬랙 봇 설명

이제 이러한 게임을 어떻게 구현했는지, 슬랙에서의 봇 구조를 설명드리겠습니다.

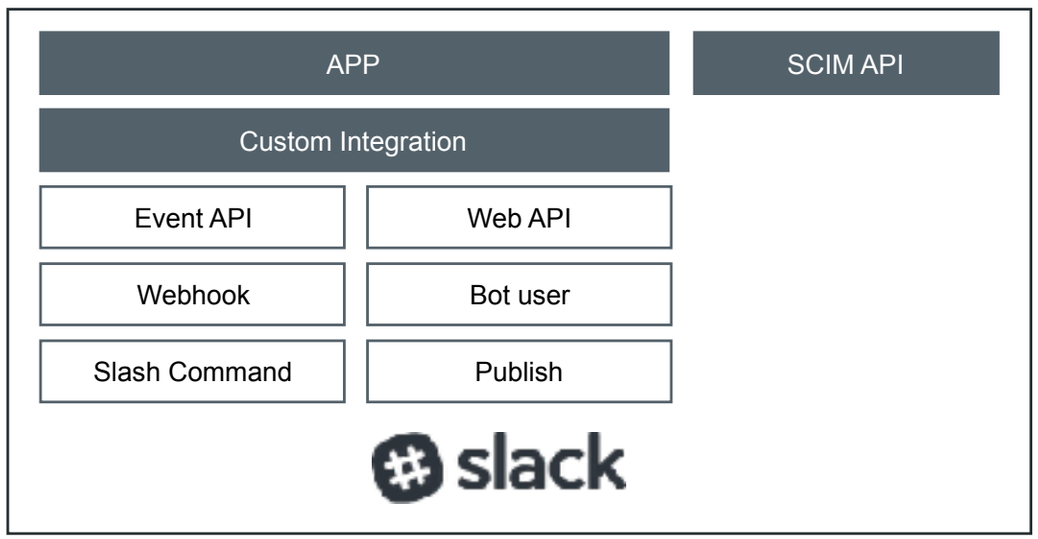
슬랙에서 봇을 사용하기



팀 단 화

우선 앞서 말한것과 같이, 슬랙은 '팀'단위의 채팅 플렛폼입니다. 팀 내에서 목적에 따라 그룹단위로 '채널'이란 곳에서 채팅이 이루어 지게 됩니다. 채널내에서 여럿이서 게임으로 경쟁하기에 최고죠. 두번째로는 슬래시 커맨드라는 기능입니다. 봇에서는 여러가지 명령어를 예약해둘 수 있고, 유저는 슬래시와 함께 명령어를 입력함으로써 봇에게 명령을 내릴 수 있습니다. 예를들어 /start라고 하면 게임시작 명령을 내리는거죠. 세번째로는 첨부파일 및 버튼입니다. 첨부파일은 다른 채팅플렛폼에도 흔히 있지만, 슬랙에서는 버튼을 통해 여러가지 액션을 취할 수 있습니다. 서핑거에서는 다중 언어를 지원하는데요. 언어를 변경하고 싶을때 버튼만 누르면 언어가 변경됩니다. 마지막으로는 봇 유저와 대화할 수 있는 기능입니다. real-time api를 통해 봇 유저와 실시간으로 대화를 할 수 있는 기능입니다. surfinger에서는 이 기능을 이용해 게임에 필요한 여러가지 데이터를 받았습니다.

슬랙 봇 구조



앞서 봤던것은 슬랙에서 봇을 어떻게 사용하는지고, 개발적인 측면에서 본다면 슬랙은 다음과같은 Event API Web API WebHook Bot User Slash Command 가 있습니다., 이번에 SurFinger를 개발을 하면서 6개 모두 사용해 보았습니다.

SurFinger 데모

이제 SurFinger의 데모를 하면서 기능들을 설명해드리겠습니다.

/helpgame 커맨드 입니다.helpgame 커맨드에서는 surfinger에서 사용할 수 있는 모든 명령어들에 대한 설명이 자세히 적혀 있습니다.

/lang 커맨드 입니다. 이 커맨드에서는 서핑거에서 제공하는 영어와 한국어를 자유자재로 바꿀 수 있습니다. 영어와 한국어 중 원하는 언어를 선택하게되면 그 언어로 변경됩니다.

/start 커맨드 입니다. start 커맨드는 2가지 모드가 있습니다. 한게임만 하는 일반모드와 여러판을 플레이하는 라운드 모드가 있습니다. 게임을 시작하게 되면 랜덤하게 미션이 주어지며 주어진 제시어를 제한시간내에 다른 유저들과 경쟁하며 입력하게되고, 제한시간이 종료되면 정확도 타속 그리고 그 둘을 기반으로 계산된 점수가 부여됩니다. 이렇게 라운드모드에선 지정한 라운드 수만큼 여러게임이 플레이되게 됩니다. 모든 라운드가 종료되게 되면 참여했던 유저들의 기록 총점이 표시됩니다.

다음은 야심차게 준비한 king of the keyboard 모드입니다. 이 모드는 한명이 남을때까지 싸우는 서바이벌모드입니다. 모드가 시작하게되면 참여자를 받습니다. 게임이 시작하게 되면 라운드가 진행되며 매 라운드마다 한명씩 줄어들게됩니다. 이렇게 계속 플레이하여 최종적으로 한명이 남을때까지 서로 경쟁하게되고 마지막 남은 한명은 King of the Keybaord의 칭호를 얻게 됩니다.

/myscore 커맨드입니다. 채널내에서 플레이했던 나의 기록을 보여주고 내가 몇등을 했는지 알려줍니다.

/rank 커맨드입니다. 채널내에 있는 모든 유저들의 최근 기록들을 기반으로 랭킹을 보여줍니다.

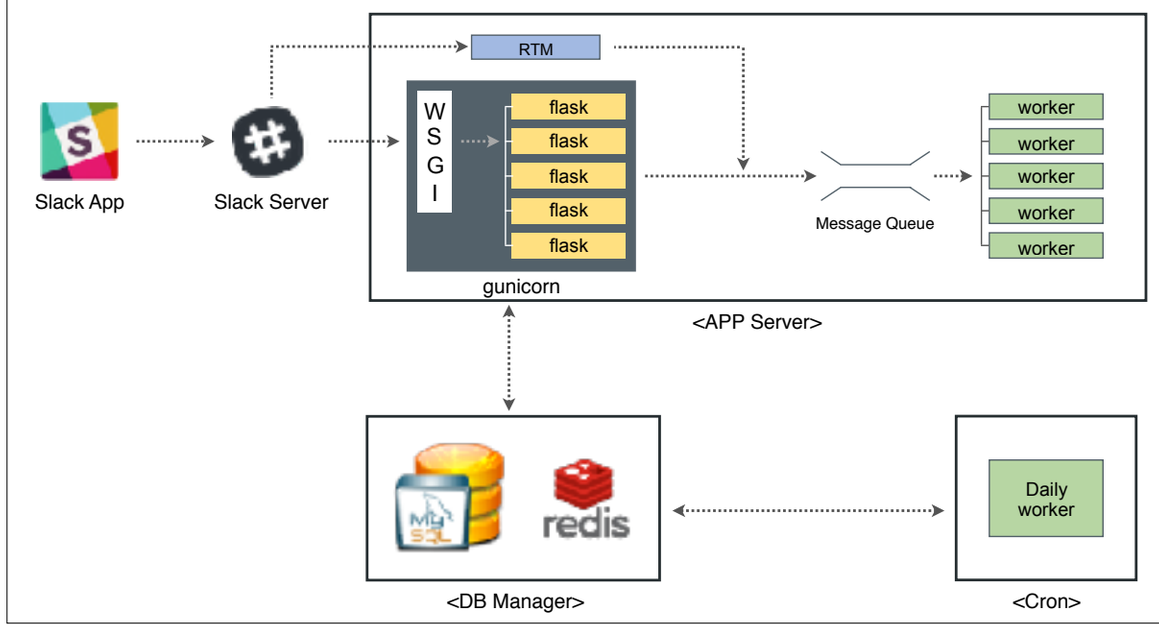
/score 커맨드 입니다. 채널내에서 플레이됬던 상위 기록들을 보여주고, 어떤 문장이 플레이됬는지 보여줍니다.

마지막으로 /badge 커맨드입니다. 특정 업적을 달성하게 되면 보상해주는 '뱃지'기능입니다. 팀 내에서 여러 유저들이 다양한 뱃지를 수집할 수 있습니다.

구현된 내용

이제 구현된 내용에 대해 자세히 설명해 드리겠습니다.

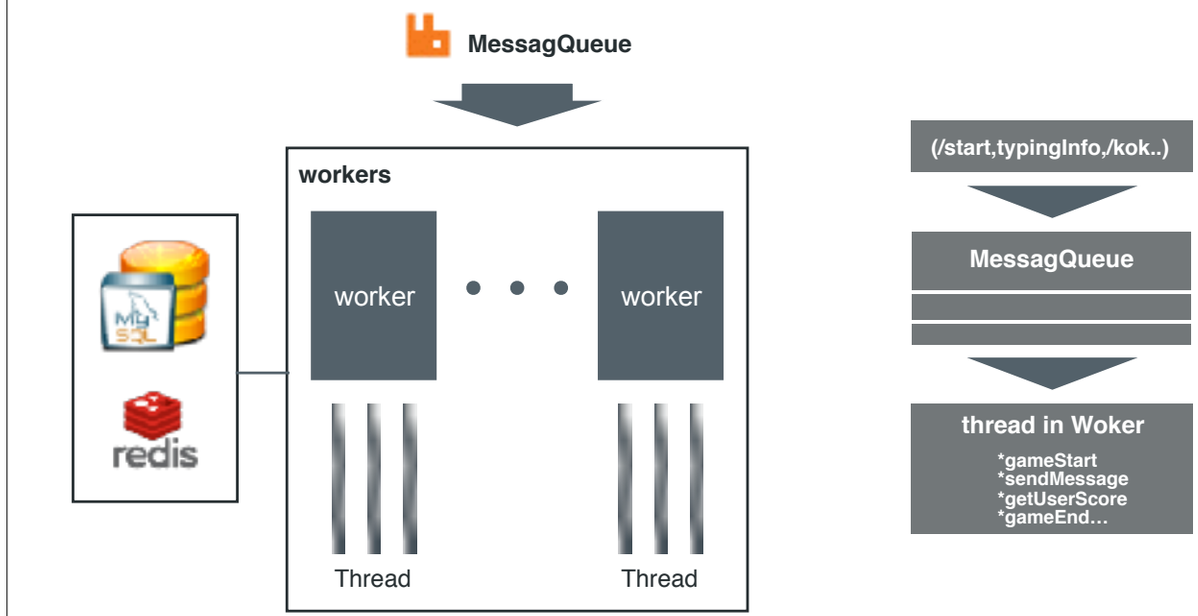
내부 아키텍쳐



전체적인 아키텍쳐입니다.(클릭) 유저의 슬랙 앱에선 슬랙 서버로만 통신을 합니다. (클릭) 저희는 OAuth를 통해서 인증받은 후 허가된 권한으로 받을수 있는 정보들, 예를들어 slash command와 같은 정보들을 http 프로토콜 방식으로 받습니다.(클릭) 그 뒤 작업들을 비동기로 처리하기 위해 amqp방식으로 message queue를 통해 worker에게 전달해 worker에서 처리합니다. (클릭) 또 게임에 실시간으로 필요한 데이터들은 Real Time Messaging 이라 불리는 socket api를 통해 받습니다. (클릭) 게임에 들어가는 모든 데이터들은 db pool과 메모리상에 저장하는 redis를 이용해 저장하고 관리합니다. (click)뿐만아니라 매일 돌아가면서 문제의 난이도나 유저의 활성도를 체크하는 cron job도 있습니다.

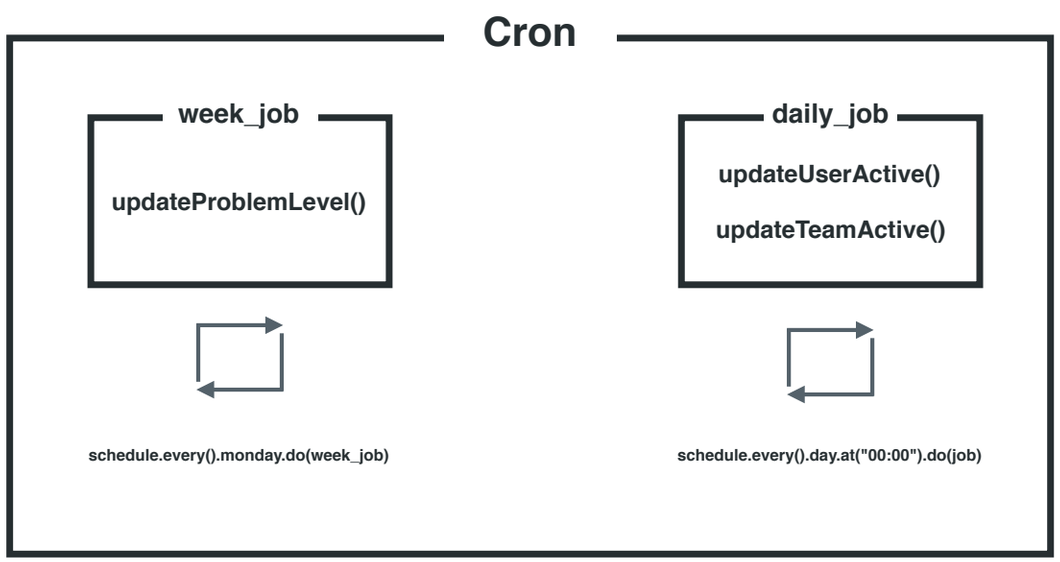
이제 모듈별로 자세히 설명해 드리겠습니다.

다른 채널내의 동시게임에 대한 이슈



다음은 worker부분을 자세히 설명해 드리겠습니다. 서로 다른 여러개의 채널에서 여러개의 게임을 플레이하기위해 여러 작업을 동시에 처리할 필요가 있었습니다. 여러개의 worker 프로세스를 만들었고, 앞서 설명드린것처럼 MQ에서 작업을 받아들이고, 워커들이 해당 작업을 워커 내부의 multi-thread에서 처리합니다. db에 굳이 저장할 필요가 없는 휘발성 데이터인 게임 status나 기타 정보들은 redis를 통해 메모리 상에서 관리해 빠르게 접근할 수 있도록 만들었습니다. 이렇게 해서 다른 채널 내의 동시게임에 대한 이슈를 해결하였습니다.

CRON 작업

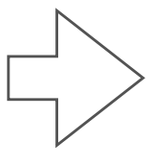


다음은 cron를 자세히 설명해드리겟습니다. cron에서는 week_job과 daily_job 이 있습니다. 게임에 사용될 문제를 웹에서 긁어왔습니다. 하지만 이러한 데이터들의 난이도를 정하는게 어려웠습니다. week_job에서는 매주 월요일마다 실행되면서 문제레벨을 유저들의 플레이 기록을 바탕으로 유저의 정확도가 낮은 순서대로 난이도를 어렵게 업데이트합니다. daily_job에서는 유저와 팀의 활성도를 매일 체크하며 7일 이상 사용을 하지 않아 비활성화됫다고 판단되는 유저들에게 푸시를 보내는 등의 작업을 합니다. 실제로 푸시를 보낸 후에는 활성화 유저가 늘어난 것을 확인 할 수 있었습니다.

정확도 계산

<기존 타자게임들의 정확도>

나는 오늘도 코딩을 한다. (100%)
나는 오늘 도 코딩을 한다. (33%)
나는 오늚도 코딩을 한다. (93%)
나는 오늘ㄷ오 코딩을 한다. (66%)



<SurFinger의 정확도>

나는 오늘도 코딩을 한다. (100%)
나는 오늘 도 코딩을 한다. (96%)
나는 오늚도 코딩을 한다. (96%)
나는 오늘ㄷ오 코딩을 한다. (96%)

다음으로는 정확도를 계산하기 위한 로직입니다. 기존의 타자게임에서 정확도를 판단하는 로직은 다음과같이 중간에 띄어쓰기가 들어가면 뒷부분이 전부 오류가 났습니다. 또한 글자 내에서 자소 하나만 틀려도 해당 글자는 전부 틀린걸로 판단했습니다. 글자에 잘못된 자소가 들어가서 글자가 틀어져 버리는 경우에도 뒷부분이 전부 틀려졌습니다. 하지만 SurFinger에서는 틀린 부분만 감점을 하고, 글자 중 자소 하나만 틀리는 경우에는 부분점수를 주고 마찬가지로 잘못된 자소가 껴서 틀어져 버리는 경우에도 틀린부분만 감점을 하였습니다. 이걸 어떻게 구했을까요?

정확도 계산

제시어 나는 오늘도 코딩을 한다.

입력어 나는 오늘ㄷ소 코드을 한다.

ㅇㅗㄴㅡㄹㄷㅗ ㅋㅗㄷㅣㅇㅇㅡㄹ ㅎㅏㄴㄷㅏ.

↓ 1글자 추가(ㅅ)

ㅇㅗㄴㅡㄹㄷㅅㅗ ㅋㅗㄷㅣㅇㅇㅡㄹ ㅎㅏㄴㄷㅏ.

↓ 1글자 변경(ㅣ -> ㅡ)

ㅇㅗㄴㅡㄹㄷㅅㅗ ㅋㅗㄷㅡㅇㅇㅡㄹ ㅎㅏㄴㄷㅏ.

↓ 1글자 삭제(ㅇ)

ㅇㅗㄴㅡㄹㄷㅅㅗ ㅋㅗㄷㅡㅇㅡㄹ ㅎㅏㄴㄷㅏ.

문장의 차이(거리) = 1추가 + 1삭제 + 1변경 = 3 = 문장의 차이(거리)

(전체글자수 - 문장의 거리) / 전체글자수 = 정확도

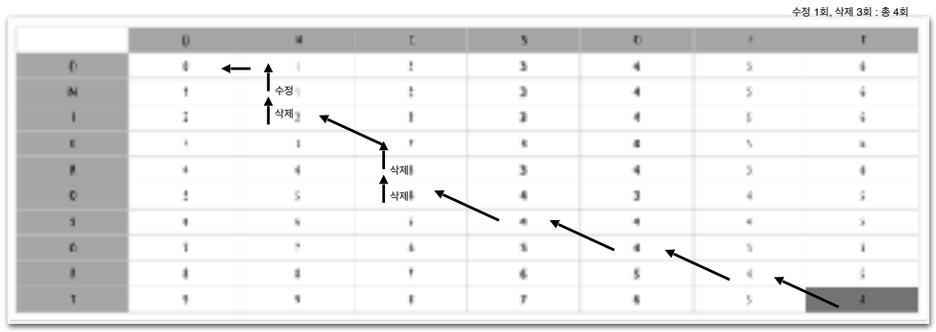
기존방식 : 정확도 **35%** 문장의 차이를 이용한 방식 : 정확도 **89%**

저희는 이렇게 제시어와 입력어가 있을때, 입력어의 정확도를 계산하기 위해 두 문장이 얼마나 차이가 나는지, 즉 제시어에서 입력어로 바뀌려면 몇번을 수정해야하는지를 계산해 봤습니다. 왼쪽 예시중 일부를 떼서 한번 계산해 보면, 우선 글자의 자소를 분리합니다. 제시어에서 입력어로 변경되려면 우선 'ㅅ'이라는 글자가 추가되고 'ㅣ' 가 'ㅡ'로 변경되어야하며 'ㅇ'이 삭제되어야합니다. 이와같은 과정은 입력어가 제시어가 되려면 3번의 수정이 필요하고 저희는 이것을 문장의 차이 즉 두 문장의 거리라고 판단하였습니다. 그렇게해서 정확도를 전체 글자수와 문장의 거리를 이용해서 구했고 기존방식에서는 위와같은 '3'번의 실수가 있는 입력어를 35%라고 판단했지만 새로운 방식에서는 89%의 보다 신빙성 있는 정확도가 나오게 됩니다.

정확도 계산

NCSOFT

MICROSOFT



Dynamic programming을 기반으로 한 Edit Distance Algorithm

$$d_{ij} = \begin{cases} d_{i-1,j-1} \\ \min \begin{cases} d_{i-1,j} + w_{\mathrm{del}}(b_i) \\ d_{i,j-1} + w_{\mathrm{ins}}(a_j) \\ d_{i-1,j-1} + w_{\mathrm{sub}}(a_j, b_i) \end{cases} \end{cases}$$

여기서 잠깐 두 문장의 거리를 어떻게 구했나 살펴보자면 Dynamic programming 기반의 Edit distance algorithm을 사용하였습니다.
다이나믹 식을 간단히 설명하자면 문장의 각 글자가 일치하면 거리가 유지되고, 두 문장의 글자가 다르면 앞서 사용된 거리중 가장 짧은것을 추가로 사용합니다.

이렇게 힘~~~들게 정확도를 계산을 해놧는데
저희 타자게임엔 치명적인 약점이 있습니다.
그 약점이 무엇일까요?
바로 붙혀넣기입니다.

붙여넣기 방지

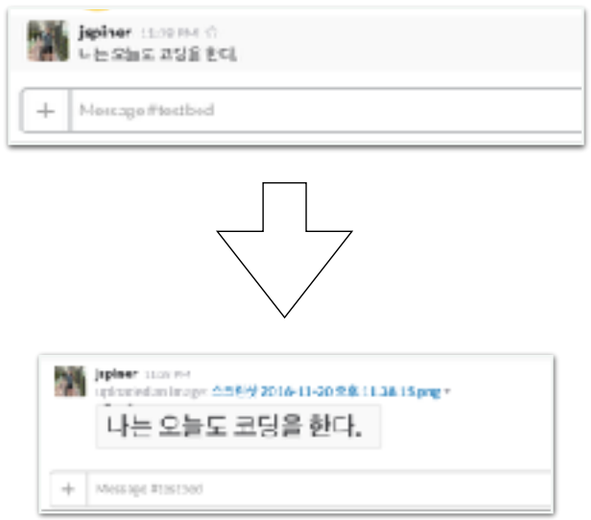
일일이 치는것보다 **붙여넣는게** 훨씬빠르다.
어떻게 막을수있을까?

정확도 계산을 만들고 커피내기를 해봤는데, 태우형이 '붙여넣기'를 통해서 이기는 일이 발생하였습니다.

저희는 커피를 사주면서 아, 이건 막아야 겠구나. 이렇게 간단한 부정행위가 허용되면 게임이 의미가 없어지겠구나 생각했습니다.

그래서 붙여넣기를 막을 방법을 고민해봤습니다.

붙여넣기 방지



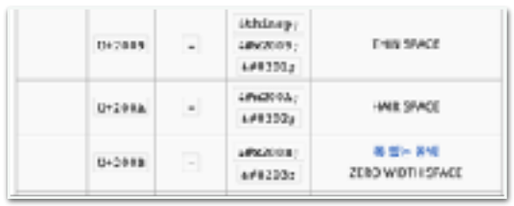
“이미지 생성에 시간 소요”

“이미지 로드에 시간 소요”

“UI적으로 어색함”

처음엔 제시어를 ‘사진’으로 만들어 채팅방에 보내는것이었습니다. 간단히 구현이 가능했지만 이미지 생성에 1차적으로 시간이 소요되고, 이미지를 불러와야하기에 유저입장에선 2차적으로 시간이 소요됬습니다. 뿐만아니라 ‘첨부파일’의 형태로 추가되는거라 어색했습니다.

붙여넣기 방지



< Zero Width Space >

그래서 유니코드중 ‘zero width space’라는것을 이용하였습니다. 이는 유니코드중 공백이지만 여백이 없는 공백입니다. 이걸 교묘하게 문장의 글자 사이사이에 넣어서 유저에게 보내면 붙여넣기를 할때 해당 글자도 함께 복사가 되고 붙여넣기를 판단할 수 있게 되었습니다. 뿐만아니라 여백이 없는 공백이기에 기존에 글자로 보내던것과 똑같아 유저입장에서도 보기 좋았습니다.

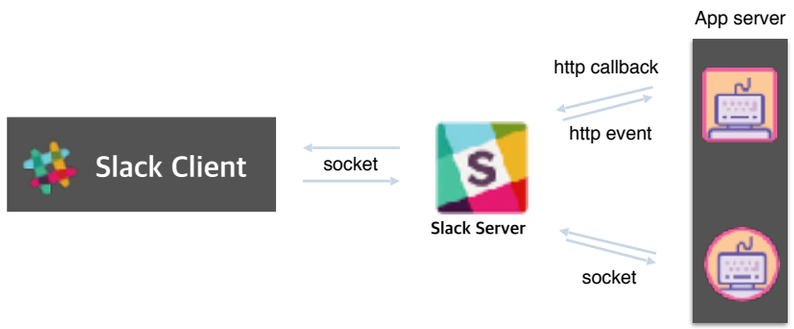
실시간 처리

http 방식으로 모든 이벤트를 처리하면 쉽고 직관적이다.
하지만 **느리다**

(Event API 최대 3~7s, http callback 약 300ms, 게임 하기에 치명적)

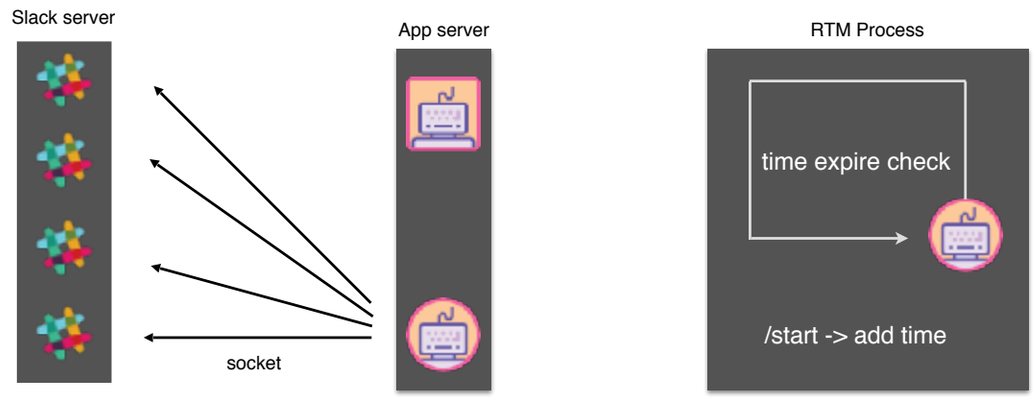
다음으로는 실시간 처리에 대해 설명해드리겠습니다. 처음엔 http로 모~든 이벤트를 받아서 처리했습니다. 하지만 슬랙 서버를 거쳐 저희 서버로 오는 Event API는 최대 3~7초 까지 소요되었고 해당 이벤트를 내부에서 처리해 http callback을 보내면 약 300ms가 소요되었습니다. 얼핏보면 느린거같진 않지만, 게임을 하기에는 치명적이라고 판단했습니다.

실시간 처리



그래서 저희는 과감하 http로 짜던부분을 socket으로 변경하였습니다. (클릭) 슬랙서버에서 데이터를 처리해서 (클릭) 실시간성이 필요한 게임 기록, 타이핑 내역 등은 socket을 통해 실시간으로 받아오고, (클릭) 시작이나 랭킹조회등 실시간성이 필요없는 부분들은 기존방식 그대로 http를 이용해 받아오도록 설계를 변경하였습니다.

실시간 처리



또한 슬랙의 구조상 봇을 사용하는 팀이 여러개가 있을 경우엔, 그 팀 수만큼 소켓을 연결해야만 했습니다. 이는 매우 비효율적이라 생각했고 저희는 게임을 사용중인 팀에게는 그대로 소켓을 열어주고, (클릭) 일정 시간이상 게임을 플레이 하지 않는 팀에겐 (클릭) 소켓 연결을 닫아주고 (클릭) 다시 팀이 활성화되면 소켓을 연결해주는 방법으로 자원을 좀 더 효율적으로 사용하였습니다.

이를 구현하기 위해 소켓 연결을 관리해주는 프로세스에서 시간을 체크하는 쓰레드를 돌리고, 해당 쓰레드에서는 최소 사용시간만큼 대기하고 다시 expire를 확인해 프로세스를 종료하는 방식으로 구현하였습니다.

여기까지는 내부 아키텍쳐에 대한 설명이었습니다. 이제 저희가 서비스를 운영하기 위해 개발한것들을 보여드리겠습니다.

AdminPage

TodayInstall/play

TodayInActiveTeam/User

Send All Notification

“유저/게임 활성화 모니터링을 통한 시스템 확장 준비”

“이벤트 등 내부 액션에 대한 사용자 피드 확인”

저희가 만든 관리자페이지입니다.
저희 관리자페이지는 단순히 db에 있는 내용을 보는것이 아니라 당일 플레이수, 비활성화된 유저들 등 운영에 필요한 다양힌 지표들을 볼 수 있도록 구현해 추후 시스템 확장을 대비하였고, 전체에게 이벤트나 공지를 보내는 등의 푸시기능 도 구현되어 있습니다. 실제로 비활성화 유저가 많을날 홍보메세지를 푸시해봤더니 실제로 비활성 유저가 줄어들고 플레이수가 늘어난 것을 볼 수 있었습니다.

LandingPage

랜딩페이지입니다. 유저가 어떻게 저희 앱을 보다 쉽게 접근하고 쉽게 설치할 수 있을까 고민을 하다가, landing page를 만들어 다양한 사용법을 제공하고 쉽게 설치 할 수 있도록 만들면 유저들이 더 편하지 않을까 생각해, 저희 봇에 대한 설명과 게임의 플레이 방법, 설치 버튼을 넣어 landing page를 만들었습니다.그 결과 기존엔 링크를 통해 설치하던 유저들이 랜딩페이지를 만든 후 설치율이 올라갔습니다.

그 외 개발된 봇들

◀ 정보 알림 봇

기존엔 새로운 설치나
각종 운영에 필요한 지표들을
관리자 페이지에서 확인하거나
db에서 직접 확인을 했다.
너무나 **불편**했다.

◀ 회의 봇

회의봇 개발이 중단된 후
프로젝트 남은 일정
확인이나 회의실 확인등
프로젝트 정보를 알려주는 봇이
필요했다.

▶ 에러 봇

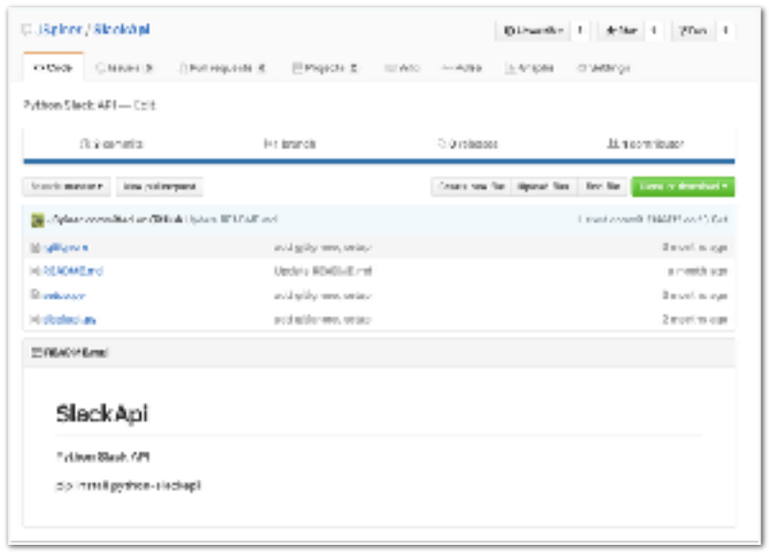
에러가 계속 나고 있던
로직이 있는데 에러로그를
보기 전까지 에러가 난지
모르고있었다.
서비스 운영에 매우 **치명적**이었다.

▶ 테스터 봇

멀티 프로세싱이나 여러 미션 등을
테스트하기 위해 일일히
테스트를 했으나 너무나 **불편**했고,
여러명이서 참여해야하는 조건을
달성하는것이 **불가능**했다.

그 외에 운영을 위해 개발한 여러가지 봇들입니다. 기존에 db나 관리자페이지에서 조회해서 보던 새로운 설치수나 특정 지표들을 조건을 만족하면 알려주는 ‘정보 알림 봇’을 만들었고, 프로젝트 진행을 위해 매일매일 남은 일정 및 회의실 정보 등을 알려주는 ‘회의 봇’을 만들었으며 지속적으로 서버를 health check를 하고, 에러가 날 경우 에러 내역을 알려주는 에러봇을 만들었습니다. 마지막으로는 여러 채널에서 동시에 플레이하거나 여러명이 참여해야하는 다양한 조건들을 테스트하기 위해 자동적으로 오타도 내고 빨리도 내고 늦게내기도 하는 다양한 환경으로 테스트하는 테스터 봇도 만들었습니다. 회의봇은 개발된 후 다른 팀에서도 사용하고 싶다고 러브콜이 와, 배포했으며 에러봇과 테스터봇은 다른 봇 개발자들을 위해 상용화 해도 많이 인기가 있을것 같습니다.

SlackApi



slack

python

마지막으로 개발을 하면서 slack api를 래핑해주는 라이브러리를 만들었는데 생각보다 편리한 것 같아 github에 오픈소스로 공개하였으며, python-pip를 통해 누구나 쉽게 사용할 수 있도록 배포하였습니다. 아직은 저희팀만 사용하고 있지만 추후엔 슬랙에서 추천하는 라이브러리로 등록되는것이 목표입니다.

성과 및 향후 진행 방향

마지막으로 저희의 성과 및 향후 진행 방향을 설명해드리고 마치겠습니다.

성과를 위한 홍보

빼빼로데이 이벤트 진행

IT관련 페이지에 홍보

저희는 출시 후 초기 유저를 모으기 위해 홍보를 진행하였습니다. 페이스북페이지를 개설하고 다양한 이벤트를 하는것과 동시에 IT 관련 페이지에 홍보하는 등 다양한 방법으로 홍보를 하였습니다.

통계자료

설치한 팀 수

32

총 플레이 횟수

660

DAILYHOTEL
펜타시큐리티
WantedX
Add2paper

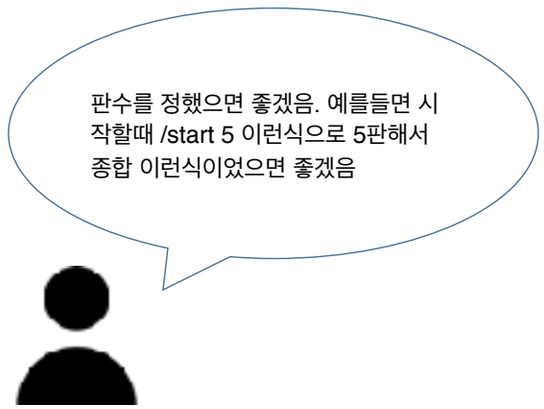
평균 주당 330회 플레이
(11.6 ~ 11.23)

그 결과 32팀이 설치를 하였고 배포한 2주간 총 660회 플레이, 즉 한주에 330회 플레이를 하였습니다. 뿐만 아니라 DailyHotel 이나 펜타시큐리티, WantedX 애드투페이퍼 등 유명한 팀에서도 설치하였습니다.

피드백

이렇게 성공적으로 배포한 것으로 그치지 않고, 게임을 일정 판 이상 플레이 하게 되면 뱃지 보상과 함께 '리뷰' 페이지로 사용자를 유도하도록 하여 구글 설문을 통해 게임에 대한 피드백을 받았습니다.

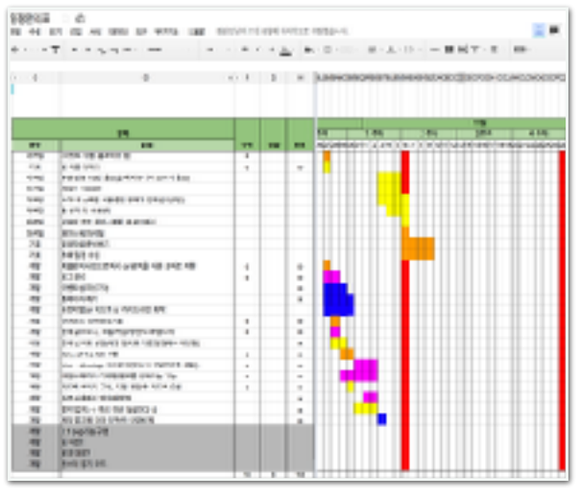
피드백



/start + [game num]
의 형태를 통해서 정해진 숫자만큼 게임플레이가 가능합니다!

피드백 중 게임을 시작할때 플레이 횟수를 정해 여러판을 한번에 했으면 좋겠다는 요청이 있었고, 그 피드백을 받아들여 새로운 모드인 ‘라운드 모드’를 만들어 릴리즈 하였습니다. 그 결과 유저들의 플레이 횟수가 늘어났고 일반모드보다 라운드 모드를 많이 플레이 하는 것을 볼 수 있었습니다.

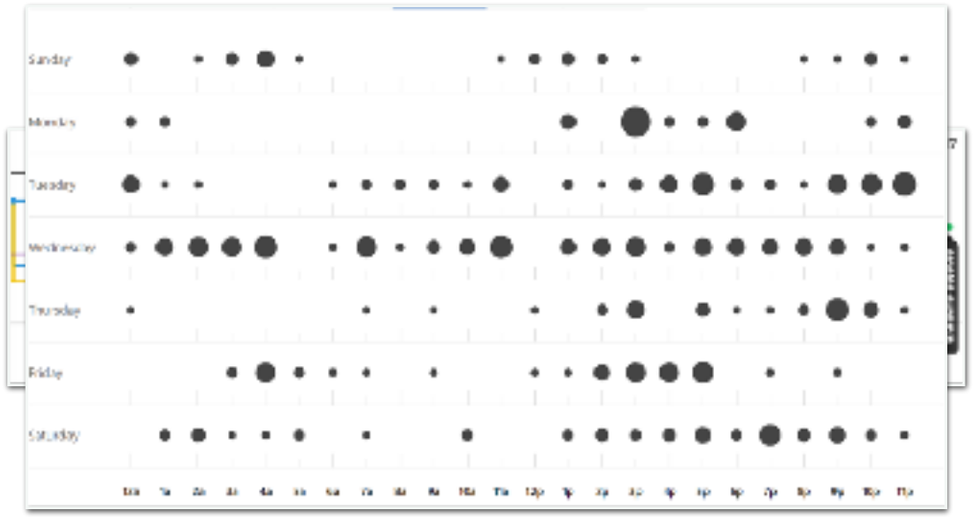
프로젝트 관리



구글 docs와 Github의 이슈와 Pull requests를 통한 프로젝트 관리를 진행

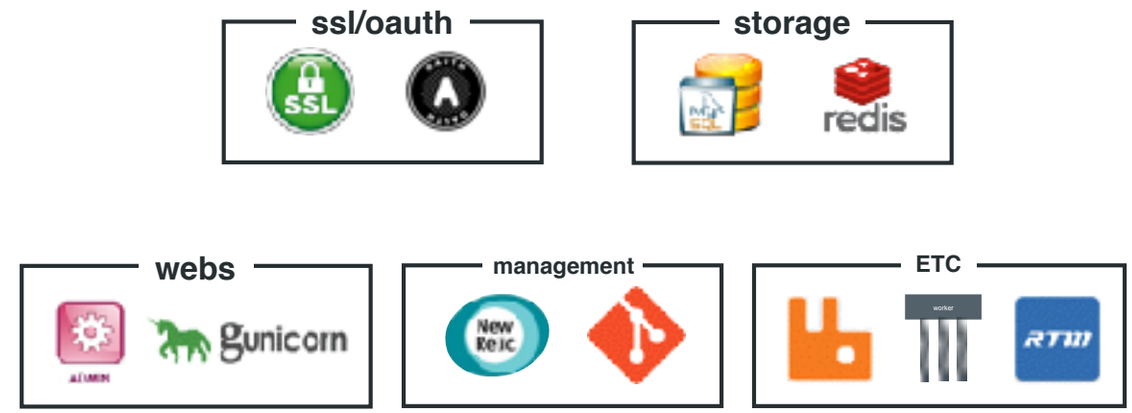
프로젝트를 관리할 때 구글독스를 이용해 간트차트를 만들어 일정관리를 하였고, 깃허브의 이슈를 만들어 그것을 티켓으로 팀원 개개인이 assign해가며 프로젝트를 진행했습니다.

프로젝트 진행



또한 브랜치를 나눠 작업하고 pull request를 통해 코드리뷰를 한 뒤 머지하는 방식으로 진행을 하였습니다. 펀치카드로 볼 수 있듯이 저희는 밤낮 가리지 않고, 무슨 요일이던지 열심히 작업하였습니다.

Develop Stack



향후 진행 계획

향후 진행 계획입니다. 우선 저희는 D.Camp와 AppCenter 에서 후원하는 챗봇 관련 세미나인 '봇 Talk'에 초청되어 개발한 SurFinger를 소개해 볼 수 있는 자리를 얻었습니다. 그 후에는 PVP 대전이나 로봇대전 랜덤 팀 대전과 같은 모드를 추가로 개설해보고 유저가 다른 유저와 점수를 비교하거나 조회할 수 있는 유저용 랭킹페이지를 만들어보고싶습니다.
그렇게해서 최종적으로는 단체 채팅 대회를 열어 많은 사람들이 함께 SurFinger를 재밌게 즐길 수 있으면 좋겠습니다.
~할 계획입니다. 저희는 이렇게 프로젝트를 끝내지 않고 계속 발전해 나갈 것입니다.

한일 그리고 느낀점

게임 플레이 로직
다양한 미션 모드 관리
게임 데이터 처리

“서비스를 구축하고 운영하는 지표 그리고 유저액션 및 피드백이 얼마나 소중한지, 멀티프로세스 환경이 얼마나 중요한지, 그리고 협업이란 무엇인지를 배웠습니다.”

Real-Time Messaging
멀티프로세싱
프로젝트 총괄적 관리

“프로젝트를 진행하면서 개발적인 측면 외에도 운영적인 측면의 중요성과 단순히 팩트만 생각하는 것이 아닌 이유와 의도를 생각하는것에 대해 배웠습니다.”

API 서버
관리자 페이지
크론 프로그램

“서버에 대해 아주 간략하게 알고 있었지만 이번 기회에 여러가지 툴과 도구 및 프레임워크를 통해서 개발하는 방법을 배웠으며 나아가 단순히 개발만 하는게 아니라 의도와 사고 그리고 이유있게 개발 할 수 있는 훈련을 해서 발전한게 느껴집니다!”

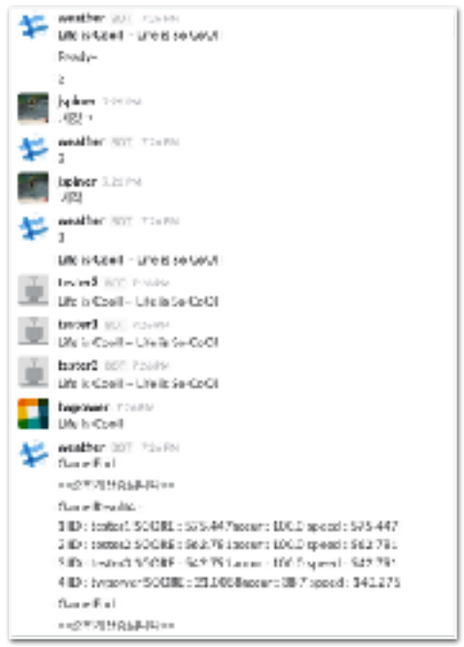
마지막으로 한일과 느낀점을 말씀드리고 마치겠습니다.

저는 rtm과 멀티프로세싱 프로젝트 총괄적 관리를 맡았으며 “프로젝트를 진행하면서 개발적인 측면 외에도 운영적인 측면의 중요성과 단순히 팩트만 생각하는 것이 아닌 이유와 의도를 생각하는 것에 대해 배웠습니다.”

Q&A

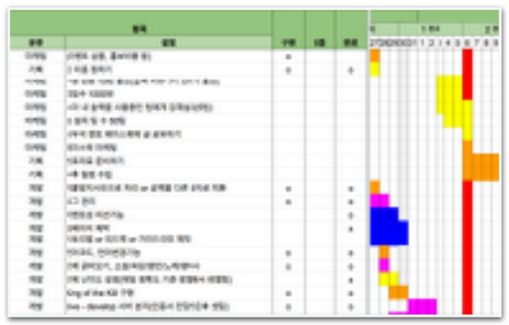
멀티프로세싱 환경 테스트

“테스트봇을 이용해 여러채널에서 동시에
플레이해보면서 멀티프로세싱 환경을 테스트해 보았다.”



다음은 저희 팀원들에 대한 설명이구요. 그 아래에 각자 맡은 역할을 간단하게 적어둔 부분입니다. 저는 ㅇㅇㅇ부분들을 맡았구요 평강이형은 ㅇㅇㅇ의 분들 그리고 태우형은 ㅇㅇㅇ 부분들을 맡아서 진행하였습니다.

협업



“**매주 3회이상** 만나서 이슈 티켓을 만들고
해당 **이슈를 assign** 해서 작업을 동시에 진행하였다.
간트차트와 구글독스를 통해 문서와 일정을 관리하였다”

다음은 저희 팀원들에 대한 설명이구요. 그 아래에 각자 맡은 역할을 간단하게 적어둔 부분입니다. 저는 ㅇㅇㅇ부분들을 맡았구요 평강이형은 ㅇㅇㅇ의 분들 그리고 태우형은 ㅇㅇㅇ 부분들을 맡아서 진행하였습니다.

사용한 오픈소스 라이센스

celery : The BSD License
flask : The BSD License
gunicorn : MIT License
slackclient : MIT License
sqlalchemy : MIT License
redisclient : The BSD License
korean : The BSD License
AdminLTE : MIT License
requests : Apache2 License

BSD 라이센스


BSD 허가서는 자유 소프트웨어 저작권의 한 가지이다. BSD 계열 소프트웨어를 포함한 많은 프로그램에서 사용한다.

BSD (Berkeley Software Distribution) 라이센스는 소프트웨어 라이선스라고도 할 수 있을 만큼 미약하다. 해당 소프트웨어는 아무나 개작할 수 있고, 수정된 것을 재한 없이 배포할 수 있다. 다만 수정물의 재배포는 의무적인 사항이 아니므로 BSD 라이선스를 갖는 프로그램은 공개하지 않아도 되는 상용 소프트웨어에서도 사용할 수 있다.

MIT 라이센스


개요 [편집]

MIT 허가서는 미국의 매사추세츠 공과대학교에서 자기 학교의 소프트웨어 공학도들을 돕기 위해 개설한 허가서이다.
MIT 허가서를 따르는 소프트웨어를 기초한 제품을 반드시 오픈 소스로 배포해야 한다는 규정이 없으며, GNU 일반 공중 허가서처럼 법적힘을 표하려는 사용자들에게 인기가 있다.

Apache2 라이센스


아파치 라이선스(Apache License) 아파치 소프트웨어 재단에서 자체적으로 만든 소프트웨어에 대한 라이선스 규정이다.

이파치 2.0 라이선스는 누구나 해당 소프트웨어에서 파생된 프로그램을 제작할 수 있으며 저작권을 양도, 전송할 수 있는 라이선스 규정을 이기된다. 이파치 라이선스에 따르면 누구든 자유롭게 아파치 소프트웨어를 기업 및 개인적 목적 혹은 상업적 목적으로 이용할 수 있으며 재배포시에는 원본 소스 코드 또는 수정한 소스 코드를 반드시 포함시켜야 하는 것은 아니고[1][2] 아파치 라이선스 버전 2.0을 포함시켜야 하며 아파치 소프트웨어 재단에 개별적 소프트웨어라는 것을 명확하게 밝혀야 한다.

다음은 저희 팀원들에 대한 설명이구요. 그 아래에 각자 맡은 역할을 간단하게 적어둔 부분입니다. 저는 ㅇㅇㅇ부분들을 맡았구요 평강이형은 ㅇㅇㅇ의 분들 그리고 태우형은 ㅇㅇㅇ 부분들을 맡아서 진행하였습니다.

Why Flask?

Light weight, fast micro framework

팀원 모두 node js로 개발을 하다가 비동기에 익숙하지 않아서 조금더 개발 효율을 높이고자 프레이워크를 찾아보던 중에 파이썬 언어가 개발하기에 편하며 플라스크가 가볍고 시작하기에 장벽이 낮다고 해서 시작하게 되었습니다.

Gunicorn Usage

Standalone WSGI Containers

There are popular servers written in Python that contain WSGI applications and serve HTTP. These servers stand alone when they run so you can proxy to them from your web server. See the section on Proxy Setups if you run into issues.

Gunicorn

Gunicorn 'Green Unicorn' is a WSGI HTTP Server for UNIX. It's a pre-fork worker model ported from Ruby's Unicorn project. It supports both eventlet and greenlet. Running a Flask application on this server is quite simple:

```
gunicorn myproject:app
```

Gunicorn provides many command line options — see gunicorn -h. For example, to run a Flask application with 4 worker processes (-w 4) binding to localhost port 4000 (-b 127.0.0.1:4000):

```
gunicorn -w 4 -b 127.0.0.1:4000 myprojectapp
```

Table Of Contents

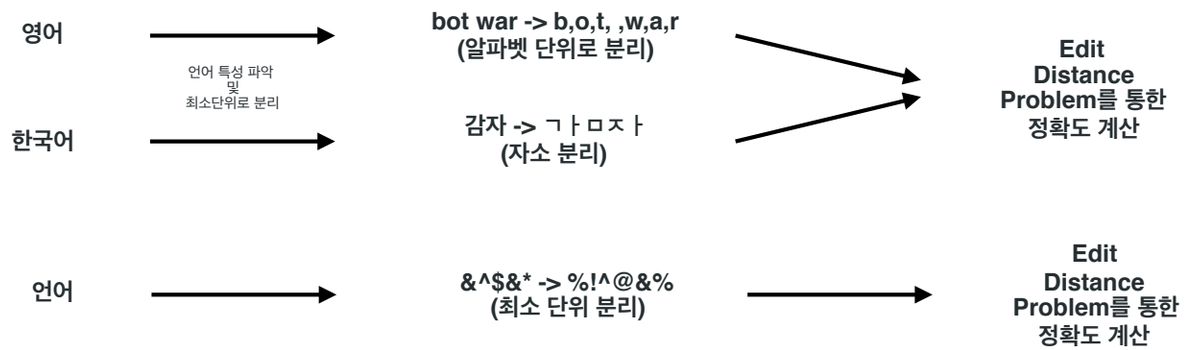
Standalone WSGI Containers
- Gunicorn
- Tornado
- Twisted Web
- Proxy Setups

Versions

development (unstable)
19.9.0 (stable)
0.17.4

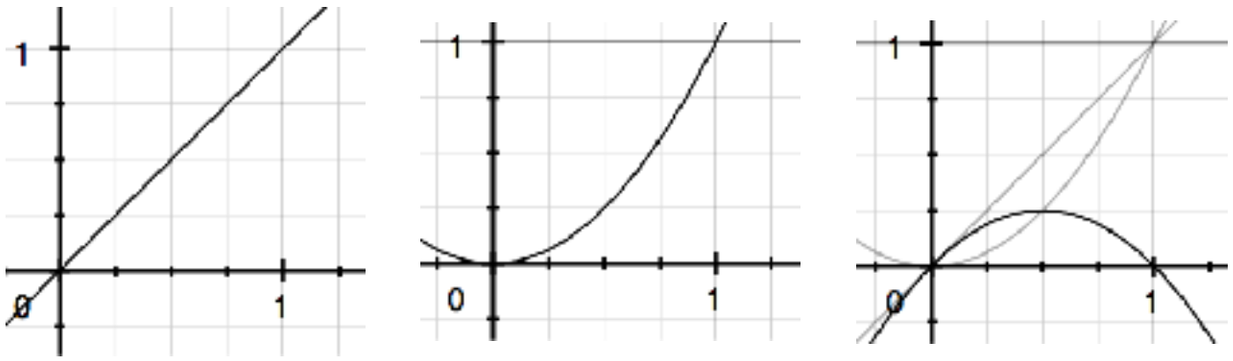
Simplest WSGI Container

다른 언어에 대한 지원 계획



각 언어의 특성을 파악한 후에
여러 국가 언어 지원 예정

속도와 정확도로 점수를 어떻게 계산했는가?



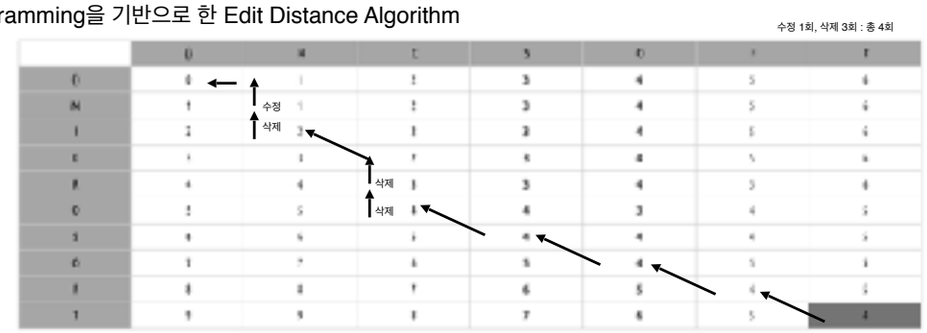
점수 = 정확도^2 * 속도

Dynamic programming을 기반으로 한 Edit Distance Algorithm

d(i,j) =

d(i-1,j-1)

min(d(i-1,j)+1, d(i,j-1)+1, d(i-1,j-1) +1)




```
def get_edit_distance(string1, string2):

    s1 = split_character(string1)
    s2 = split_character(string2)

    d = [[0 for col in range(len(s2) + 1)] for row in range(len(s1) + 1)]

    for i in range(0, len(s1) + 1):
        d[i][0] = i

    for j in range(0, len(s2) + 1):
        d[0][j] = j

    for i in range(1, len(s1) + 1):
        for j in range(1, len(s2) + 1):
            if s1[i - 1] == s2[j - 1]:
                d[i][j] = d[i - 1][j - 1]
            else:
                d[i][j] = min(d[i - 1][j] + 1, d[i][j - 1] + 1, d[i - 1][j - 1] + 1)

    return d[len(s1)][len(s2)]
```

http 방식과 socket 방식은 안전한가?