콘샐러드 3회 - 2019.03.16

시작하게 된 계기 : spot meet up //점심에 모여서 특정 주제에 대한 많은 사람들과의 대화.

(개발 팀 인원 부족-직군 별 한두명- 으로 인해 더 많은 사람들의 이야기를 듣고 싶어서.)

격월로 meep up 실시. (5월에 시작.)

순서

- two track (web, mobile) //총 3번

- fireside chat (Q&A)

- sli.do #6639 //온라인 질의응답

// 1. 코틀린 퍼즐러(mobile)

코틀린 퍼즐러 url 링크

https://goo.gl/ejrMtw

총 7문제, 각 1~2분

- 디버깅 할 때, 당연하다고 생각했던 당연하지 않은 문제들.

1.My BankAccount

https://goo.gl/forms/xHNWD7om9jv8HhB22

데이터 클래스. 부모의 함수를 사용하지 않고 재정의 하는 문제가 있었음.

(부모의 함수를 사용하겠다고 명시적으로 사용, )

2. Engineer’s lover

https://goo.gl/forms/GUYp2nCWQrBOmGGH3

"?.let"

kotlin null check. (?.)

let, run, also, apply : expression(return 값이 있다. statement의 반댓말) / scope함수들(let, run, also, apply)의 차이점을 생각하면서 사용.

-this/param 사용 유무를 생각하자.

구문?:구문 -> 왼쪽 구문이 null이 아니면 왼쪽 return 내용을, 아니면 오른쪽 구문이.

3.Easy Book

https://goo.gl/forms/IHGmoSm24kjo7ZX43

!= 와 ?: 의 연산자 우선순위.

!= 의 우선순위가 더 높기 때문에, 왼쪽 구문 != 오른쪽 구문 이 되어서 true.

4.No account, No password

https://goo.gl/forms/le7ZsRtk8dPzI0M62

null check에 대한 부분.

account?.toString() : null

password.toString() : "null"

\* any 확장함수 toString으로 인함.

5.My name is

https://goo.gl/forms/gpQlkGkcNdlB6SRR2

User.newInstance("james") : complie error. 객체에서 바로 사용할 수 없다.

\*extension 함수

class에서 바로 접근하고 싶을 땐 Companion object로 만들어서 사용.

6.Memo

https://goo.gl/forms/VFZ0uF7nCvTDFGVO2

jUnit.assert != kotlin.assert

kotlin assert 함수 : assertion.ENABLED가 true일 때만 확인됨.

\*실패하는 test code를 먼저 작성하라.

7.Be our guest

https://goo.gl/forms/M60wSmqXy5xLO2AS2

6 6

6 7 7

6 8 7

변수가 어떻게 치환되는지.

sizeValue : final int // 변수 선언되면서 6

sizeGet : final function // 함수 호출될때마다 +1 (custom getter)

sizeLazy : final lazy // 처음 사용하는 순간에 초기화. 이후 고정

-kotlin val == same as java final

질문 : 채용 형태는(신입/경력) ?

답변 : 안드로이드 팀 5명. 경력은 중요하지 않다. 우리 팀에 녹아들 수 있는 사람을 원한다.

// 2. 뱅크샐러드 파이썬맛 레시피

- 백엔드 팀의 파이썬 이야기

>환경

lean 한 학습. 빠른 개선과 배포 (빠른 피드백, 개선의 반복)

마이크로서비스 아키텍쳐

네트워크 io가 많은 msa 환경, 적극적인 비동기 활용

-gevent 경량 스레딩 / 이벤트 루프 비동기 방식-선택)

--일반적인 익숙한 패턴

--나이스한 문법

-python 3.7 이후 버전

>패턴

-애플리케이션

--sanic 비동기 패턴 (aiohttp, flask 대신 사용)

더 빠르고 간결한 문법, 확장된 기능

--클린아키텍쳐 : 제품 관점 / 협업 관점 / 개발 환경 관점에서 많은 이득이.

---도메인을 중심으로. 공통된 구조를 통해 유지보수성 증가. 일관성 있는 코드 확장된

--도커

---멀티 스테이지 빌드. 1/3정도로 절약

-테스트

--pytest lib (unittest + nose보다 더 좋다고 판단)

--coverage : 목표가 아닌 상태로 참조

--lint, style(isort, pylint, )

--git hook을 통한 테스트 자동화 (오늘 할 일을 내일로 미루지 말자. 짧은 피드백 주기를 위해.)

>협업

-코드 컨벤션. 사실 협업의 영역

-Rainist/styleguide : 전원 합의를 거친 코드 컨벤션

-Rainist/python (cookiecutter)

-gitHub 같이 잘 쓰기

--issue\_template.md : 맥락 파악, 레거시화, 의사소통 비용 감소

--pull\_request\_template.md : 일정, 이슈, 변경사항 등 체크.

--이슈 라벨링.

--코드 리뷰 : 500줄 이내로 하자. (500줄 미만일 때 더 활발하게 리뷰 할 수 있었다.)

---리뷰어 정보 제공, 컨텍스트 공유를 위한 자리 마련(코드 리뷰 미팅 요청)

Q&A

배포는 어떻게 ?

개발환경 ? 각자 기호(vs code, sublime, pycham)

개발/디자인 이슈 트래킹 ? 개발은 gitHub, QA는 지라

db ? mongodb / mysql

python 3.7 은 호환 안되는 것들이 많던데..호환은 어떻게(ex.crawlling) ? 3.7로 통일되어 있어 문제를 겪지 못함

일반 debugging과 pytest의 차이 ? 디버깅과 테스트를 분리함. pytest는 테스트. 테스트를 통과하지 못하면 디버깅 (개발자가 알아서.)

\*테스팅과 디버깅을 구분하자.

뱅크샐러드는 금융정보. 망 분리 규제에 대한 문제 및 해결방법 (해결이 안되는 부분에 대한 생각) ? 금융 규제에 대한 파악은 아직 못하였으며, fireside chat에 올려주면 좋을 듯.

// 3. 담대한 협업과 클린 코드

>야생 개발자

-협업 경험이 적은 개발자

--문과 출신 개발자. "프로그래밍, 이거 제법 문과스럽다."

--시 쓰듯 프로그래밍. 멋진 코드는 함축적인 코드

변수/함수 이름 : evt, evt1, has\_c, s, en, f ...

--우아한 코드는 적당히 어려운 코드 (대입하면서 호출하면서, 할당하면서 대입하면서 ...)

--깨끗한 코드는 짧고 예쁜 코드는

--조금 어려워도 옆 사람만 이해하면

>담대한 협업

-담대한 협업. 뱅크샐러드에서는 협업을 매우 강조한다.

--다섯가지 키워드.

--"공유와 피드백"

--- 2달의 수습, 3회의 피드백.

--개발자의 공유와 피드백 : 코드 리뷰

clean code (책) 스터디

>클린코드

-깨끗한 코드

"wtf"의 갯수로 코드를 평가

"깨끗한 코드는 잘 쓴 문장처럼 읽힌다."

-의미있는 이름

"의도를 분명하게 밝혀라"

clearKey -> compact -> compactObject(compactString, ...)

-함수와 주석

"프로그래밍은 여느 글짓기와 비슷하다."

tag&& -> !!tag&& -> optional(tag&&..) -> optional(tag, it=>...)

"주석 대신 코드"

//s는 셀렉터, en은 이벤트명, f는 리스너

evt(s,en,f)

->

add\_event\_listener = (select,event,listener)...

-오류 처리

"논리와 오류 코드를 섞지 말라."

--비즈니스 로직에 집중할 수 있게 하자.(=논리와 로깅 코드를 섞지 말자.)

>뱅샐 개발자

-멋진 코드는 설명이 필요없는 코드

-우아한 코드는 예상 가능한 코드

-깨끗한 코드는 수정이 용이한 코드

\*클린 코드는 협업을 위한 코드

협업에 대해서 팀원과 맞춰봐야 할 필요가 있다.

>TMI.클린코드 스토디

-사내 스터디. 8명

-js, ts, python, kotlin, swift,... 다양한 코드에 대해

Q&A

코드 설명 버전 :

1. 함축적인

2. 이름을 잘

3. 비즈니스 로직과 로깅 로직이 섞임

4. 로깅을 따로.

Q) 코드 내에서 주석/사용법 설명 ..

A) 주석은 지양, 전반적인 설명은 문서화.

Q) 수습2달 피드백 방법

A) 좋은점/고칠점에 대한 피드백. 다음 피드백 때는 고쳐졌는지.

Q) 개발 팀 내부의 커뮤니케이션에는 도움되지만, 기획/디자인등 다른 팀과의 의사소통은

A) 개발자가 기획에 참여할 수 있도록 권장. '각자 한명한명이 pm이 되어야 한다.'

슬랙 등으로 회의, 미팅 시간 잡아서 회의 등으로 해결.

+ 기획쪽의 depth는(prototype or 문서) ? 기획과 개발이 계속 티키타가.

Q) 이벤트 트랙킹

A) firebase 같은 곳으로 logging을 넘긴다.

Q) 각 다른 파트에 대한 언어의 정의 등이 다를텐데..(커뮤니케이션 워딩)

A) project 간 domain 지식 (강의를 통해 통일시킴) & 계속 질문.

Q) 실무 코드가 정말 클린한지를 측정하는 지표가 있는지

A) 따로 가지고 있진 않음. 경험적인 느낌. 커뮤니케이션 스킬 등이 일치되는 느낌이 든다.

// 4. fireside chat

Q) 발표자료

A) slideshare 등에 개인적으로 올리고, 뱅크샐러드 facebook에 공지. (+뱅크샐러드 유툽에 실황도.)

Q) ios 추가 채용

A) 4명. (최대 7명 예상). 거의 모든 포지션 t.o가 열려있음

Q) 신입 웹개발자 연봉

A) wanted에 공개되어있음. 신입 평가가 어렵다. 4200을 받을 수 있는 실력

Q) 4년 이하 주니어 프론트개발자에게 요구되는 기술 스택

A) typescript / react + 프로그래밍적인 감/커뮤니케이션 역량. 스펙보다는... , 단순 코딩테스트 뿐 만 아니라 협업에 필요한 역량(협업에 필요한 코딩). 기술 스택이 다르다고 마이너스가 되진 않는다.(맞으면 플러스)

Q) ios/android 코딩 테스트? 기술면접?

A) android(사전 과제/리뷰, 시간제한 라이브 과제 해결-과정 중시-), 기술면접, 컬처면접. ios도 비슷. 지원자의 경험에 따라 상황은 달라진다.

Q) 평균 연령과 경력

A) 기술팀 전체 평균(30초반. 32~33) / 3~4년. 각 파트 별 시니어 존재.

Q) backend python으로만

A) python(단순 crud), node, scala(부하 많은 부분), (golang도 썼었다.)

Q) sanic 부하문제 / http 2

A) sanic은 부하를 신경쓰지 않는다. 트래픽 처리에 대해 잘 처리되어서 사용.

Q) rainist 채용 공고가 오래 열려있음

A) 할일이 많아서 to가 늘어남. 작년대비 사용자 수 2배.

Q) 다른 회사와의 다른점

A) 커뮤니케이션 능력. 각각의 협업능력이 좋다.

Q) 핀테크 회사/은행들은 인터넷 은행

A) 인터넷 은행 안함. 금융 플랫폼으로 좀 더 .

Q) 업무 프로세스

A) 프로젝트 별 시작 시 디자이너가 기획 -> 킥오프 미팅(개발, 비즈니스 책임자 등) -> 스탠드업 미팅 -> 프로젝트 마무리 -> 회식 (아직은 작은 회사라 명문화된 프로세스라고 할 건 없다.)

Q) 배포 파이프라인 / 인프라 파이프라인

A) 배포 프로세스 (브랜치->디플로우 전략->ci->docker hub->depops deploy bot->배포)

Q) 세미나/컨퍼런스에 많이 보이는데... 발표 의무

A) 발표 의무 없음. 자원. 준비하면서 많이 배움(스스로 정리하고 싶을 때 자원)

Q) 뱅크샐러드 앱(모바일. 네이티브? 하이브리드?)

A) 틀은 네이티브. 하이브리드(웹뷰 -> 재태크 탭) 성능이 중요하면 네이티브. 실험이 필요하면 웹뷰.

Q) 백엔드/데브옵스 주니어 인터뷰 할 때 예상 질문/코딩테스트

A) 성장가능성. 그리 어려운 문제를 내진 않는다(특정 자료구조가 필요하지 않는). 어떻게 해결하는지. 역량이 있는지

Q) 떨어지고 재지원하면?

A) 재지원해서 붙으신 분들도 있음.

Q) 모바일 개발자를 뽑을 때의 과거 경력이 모바일이 아니더라도 +/- ?

A) 이력서 바이 이력서. 고과를 최대한 균형있게. 직군이 변경될 때의 퍼포먼스 때문에 사람 바이 사람.

Q) 본인이 생각하는 좋은 회사? rainist는 어떤 회사?

A) 회사가 고민하는 것(추구하는 목표)과 자신의 마음이 맞는지를 판단하자. 사람/비전/제품을 기준으로 판단.

Q) 나이 많으면 안뽑나요

A) 나이가 많은 것이 +/- 포인트가 되진 않는다.

Q) 채용 프로세스

A) 서류->전화 인터뷰->코딩 인터뷰/컬처 인터뷰

Q) 인턴은 어떤 일을 하게 되나요

A) 케바케. 데이터 내보내기 기능.

Q) 사내 스터디 문화

A) 클린코드 스터디 등, 출근 시간 전 모임.

Q) 기획자가 없이 개발하는 것의 장단점

A) 사람에 따라 다름. 개발에 있어 주체적인 의식을 가지고 애정을 가지고 말할 수 있음/개발 외의 시각을 배울 수 있다. 단점 : 시간이 많이 걸린다.

Q) 요즘 개발하고 있는 프로젝트

A) 국민연금 등의 연금보기,연금 진단 (안드로이드 오픈. ios 심사중), 재테크 탭(추천 기능-사용자를 위해. 광고 없음)

// 뱅크샐러드 페이스북

https://www.facebook.com/BanksaladTip/

// 개인적으로 느낀점

1) 서로를 부르는 호칭이 "oo님". 수평적인 조직문화로 보임

2) 신기술을 많이 도입하는 느낌은 아니었지만, 좀 더 좋은 기술력을 가지기 위해 개개인이 노력하는 걸로 느껴짐. 들어가면 많이 배울 수 있을 듯

3) 연봉 와우…