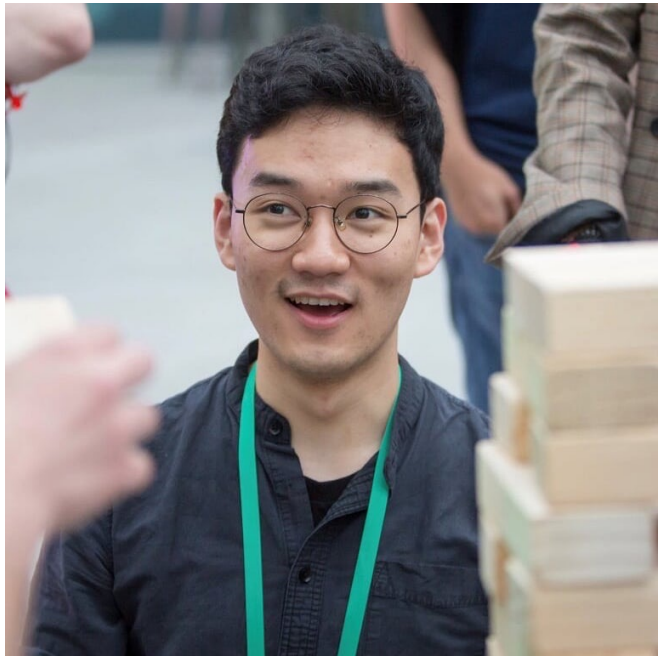
기술 문서 작성법 특강

이스트소프트 백엔드 개발자 양성 과정 오르미 1기

강사소개



계성혁

- 현) LG전자 데이터 엔지니어
- 전) (주) 더블에이치 서버 엔지니어 인턴
- KAIST 소프트웨어대학원 졸업 (지도교수: 강지훈)
- 아주대학교 소프트웨어학과 졸업

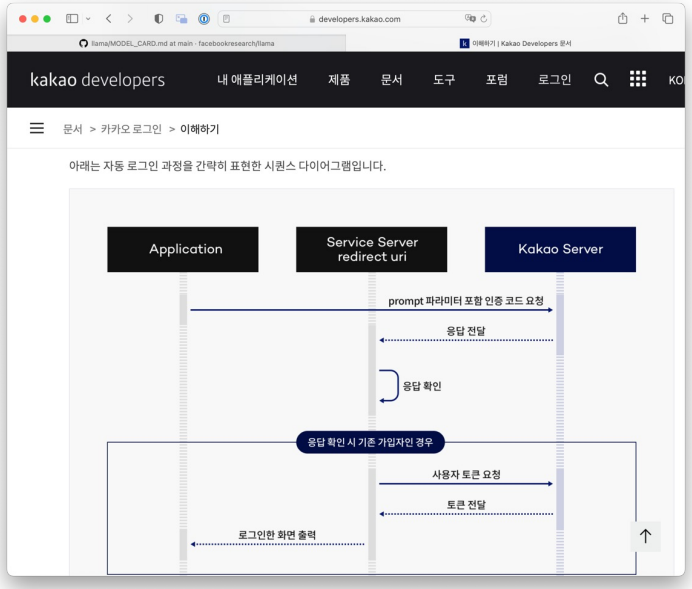
강의 시간표

시간	학습 내용
09:00 ~ 09:50	기술 문서의 필요성
10:00 ~ 10:50	소프트웨어 개발과 기술 문서
11:00 ~ 11:50	기술 문서 작성 방법 (SRS, 요구사항 명세서)
13:00 ~ 13:50	기술 문서 작성 방법 (다이어그램)
14:00 ~ 14:50	기술 문서 작성 방법 (도구를 통한 자동화)
15:00 ~ 15:50	문서화 실습 (1)
16:00 ~ 16:50	문서화 실습 (2)
17:00 ~ 17:50	팀별 결과물 발표

기술 문서?

우리가 생각할 수 있는 기술 문서들은 어떤 것들이 있을까요?

(생각보다 광범위하게 볼 수 있습니다.)



Source:이해하기 | Kakao Developers 이해하기

Llama 2: Open Foundation and Fine-Tuned Chat Models

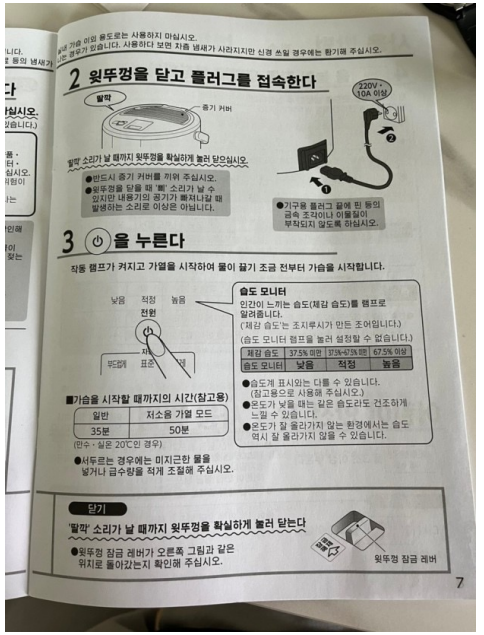
Hugo Touvron* Louis Martin† Kevin Stone†
Peter Albert Amjad Almahairi Yasmine Babaei Nikolay Bashlykov Soumya Batra
Prajjwal Bhargava Shruti Bhosale Dan Bikel Lukas Blecher Cristian Canton Ferrer Moya Chen
Guillem Cucurull David Esiobu Jude Fernandes Jeremy Fu Wenyin Fu Brian Fuller
Cynthia Gao Vedanuj Goswami Naman Goyal Anthony Hartshorn Saghar Hosseini Rui Hou
Hakan Inan Marcin Kardas Viktor Kerkez Madian Khabsa Isabel Kloumann Artem Korenev
Punit Singh Koura Marie-Anne Lachaux Thibaut Lavril Jenya Lee Diana Liskovich
Yinghai Lu Yuning Mao Xavier Martinet Todor Mihaylov Pushkar Mishra
Igor Molybog Yixin Nie Andrew Poulton Jeremy Reizenstein Rashi Rungta Kalyan Saladi
Alan Schelten Ruan Silva Eric Michael Smith Ranjan Subramanian Xiaoqing Ellen Tan Binh Tang
Ross Taylor Adina Williams Jian Xiang Kuan Puxin Xu Zheng Yan Iliyan Zarov Yuchen Zhang
Angela Fan Melanie Kambadur Sharan Narang Aurelien Rodriguez Robert Stojnic
Sergey Edunov Thomas Scialom*

GenAI, Meta

Abstract

In this work, we develop and release Llama 2, a collection of pretrained and fine-tuned large language models (LLMs) ranging in scale from 7 billion to 70 billion parameters. Our fine-tuned LLMs, called Llama 2-Chat, are optimized for dialogue use cases. Our models outperform open-source chat models on most benchmarks we tested, and based on our human evaluations for helpfulness and safety, may be a suitable substitute for closed-source models. We provide a detailed description of our approach to fine-tuning and safety improvements of Llama 2-Chat in order to enable the community to build on our work and contribute to the responsible development of LLMs.

Source: Llama 2: Open Foundation and Fine-Tuned Chat Models



Source:조지루시 사용설명서 : 네이버 블로그

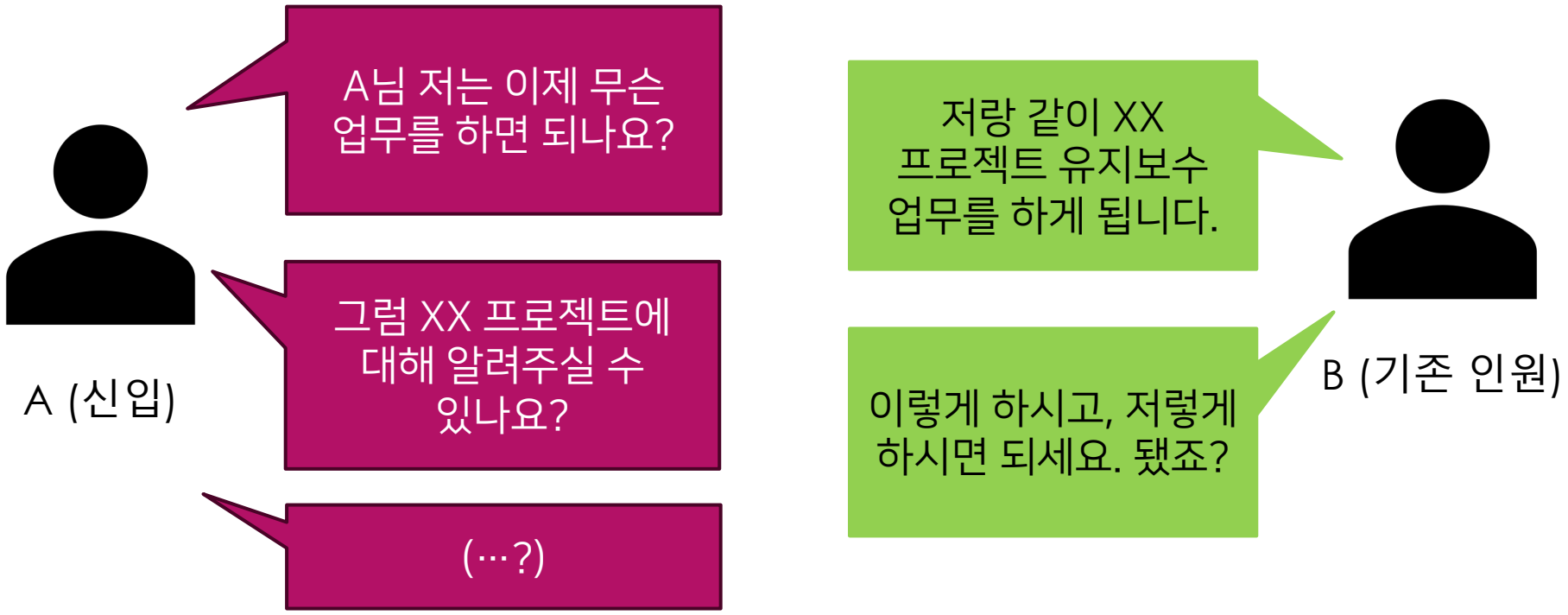
기술 문서가 왜 필요한가?

Source: 내가 뜬금없이 LG 그램 스타일 14인치를 구입한 이유..? 정말 간단한 언빡싱! - YouTube

> 아무리 좋은 기술력을 가지고 있다고 하더라도 기술이 전파될 수 없다면 그 효용은 떨어집니다.

기술 문서는 기술을 전파할 수 있다는 가치 때문에 "자산"이 될 수 있습니다.

기술 문서가 왜 필요한가? (1)

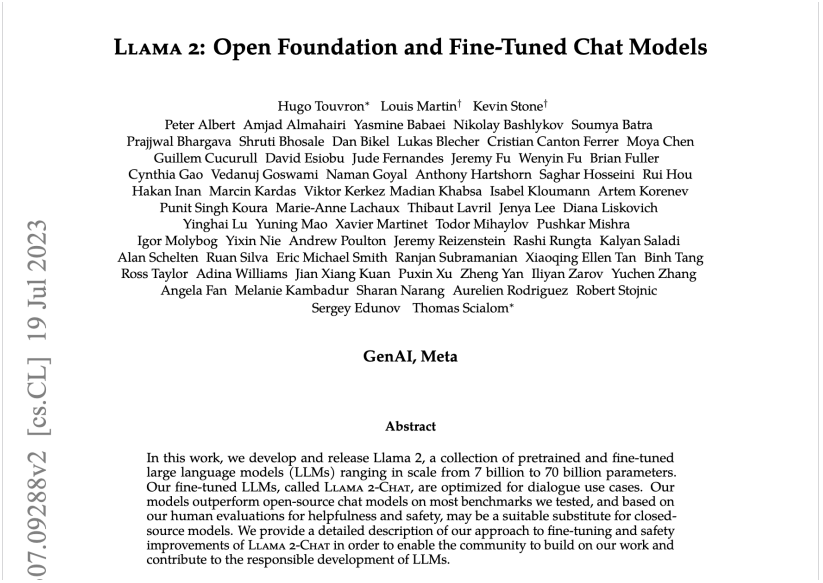


유의사항: 극단적인 예시입니다.

기술 문서가 왜 필요한가? (2)

신기술 창조의 밑거름이 됩니다.

- 예) 대학원 논문
  - 매년 나오는 새로운 논문은 학계의 집단 지성을 공개하는 장으로 공개된 자료에 기반해 새로운 발전을 일궈내는 판을 제공합니다.
- 예) 신입 사원 온보딩
  - 잘 정리된 문서는 재현 가능성(Reproducibility)를 높여 소모적인 일에 투자하는 시간을 줄여줍니다.
  - 결과적으로 더 생산적인 일에 투자할 수 있는 밑거름이 됩니다.



Llama 2: Open Foundation and Fine-Tuned Chat Models

Hugo Touvron* Louis Martin† Kevin Stone†
Peter Albert Amjad Almahairi Yasmine Babaei Nikolay Bashlykov Soumya Batra
Prajjwal Bhargava Shruti Bhosale Dan Bikel Lukas Blecher Cristian Canton Ferrer Moya Chen
Guillem Cucurull David Esiobu Jude Fernandes Jeremy Fu Wenyin Fu Brian Fuller
Cynthia Gao Vedanuj Goswami Naman Goyal Anthony Hartshorn Saghar Hosseini Rui Hou
Hakan Inan Marcin Kardas Viktor Kerkez Madian Khabsa Isabel Kloumann Artem Korenev
Punit Singh Koura Marie-Anne Lachaux Thibaut Lavril Jenya Lee Diana Liskovich
Yinghai Lu Yuning Mao Xavier Martinet Todor Mihaylov Pushkar Mishra
Igor Molybog Yixin Nie Andrew Poulton Jeremy Reizenstein Rashi Rungta Kalyan Saladi
Alan Schelten Ruan Silva Eric Michael Smith Ranjan Subramanian Xiaoqing Ellen Tan Binh Tang
Ross Taylor Adina Williams Jian Xiang Kuan Puxin Xu Zheng Yan Iliyan Zarov Yuchen Zhang
Angela Fan Melanie Kambadur Sharan Narang Aurelien Rodriguez Robert Stojnic
Sergey Edunov Thomas Scialom*

GenAI, Meta

Abstract

In this work, we develop and release Llama 2, a collection of pretrained and fine-tuned large language models (LLMs) ranging in scale from 7 billion to 70 billion parameters. Our fine-tuned LLMs, called Llama 2-Chat, are optimized for dialogue use cases. Our models outperform open-source chat models on most benchmarks we tested, and based on our human evaluations for helpfulness and safety, may be a suitable substitute for closed-source models. We provide a detailed description of our approach to fine-tuning and safety improvements of Llama 2-Chat in order to enable the community to build on our work and contribute to the responsible development of LLMs.

07.09288v2 [cs.CL] 19 Jul 2023

Source: Llama 2: Open Foundation and Fine-Tuned Chat Models

사전 과제

여러분의 집에서 주변 번화가까지 가는 방법을 서술하세요.

- 주변 번화가는 여러분이 거주하는 지역에서 사람들이 많이 몰리는 장소를 의미합니다.
- 예) 서울 – 강남, 사당 / 대전 – 둔산 / 부산 – 해운대 등

양식과 분량은 자유입니다.

사전 과제

만약 여러분들에게 이 문서들만 주어지고, 이 문서들만 참고해서 도착지까지 갈 수 있나요?

= 만약 여러분들에게 이 문서들만 주어지고, 이 문서들만 참고해서 바로 일을 시작할 수 있나요?

- 객관적인 지식을 일목요연하게 정리하는 것이 좋습니다.
- 바로 시작은 못하더라도, 남겨진 문서들은 일의 숙련도를 높이는 시간을 단축시켜줍니다.

기술 문서의 특징

- 문서를 읽는 목적이 분명하다.
- 검색하기 쉽게 써야 한다.
- 문장의 구조화가 잘 되어야 한다.
- 필요할 때만 필요한 부분만 본다.
- 난이도별로 내용을 배열한다.
- 문서를 읽으면서 혼선을 일으키지 말아야 한다.

구조적 (6하원칙, 체계적인 목차)

통일성 (용어 정의 통일, 템플릿)

- 빠른 시간에 이해하기를 원한다.
- 독자의 수준, 독자가 찾는 내용의 수준이 다양하다.
- 독자에게 알려줘야 하는 정보가 무엇인지를 분명히 알고 써야 한다.
- 요약한 내용과 상세한 내용을 함께 포함해야 한다.

간결성
(명확한 제목, 객관적인 표현)

쉬는 시간

10시에 다시 시작합니다.

소프트웨어 개발과 기술 문서

그 전에 건물을 올리는 과정을 생각해봅시다.

건물은 어떻게 만들어질까요?

- 지을 땅의 상황을 파악하고
- 어떤 건물을 지을지 정하고 (아파트? 오피스텔?)
- 도면을 그리고
- 쌓아올리고…

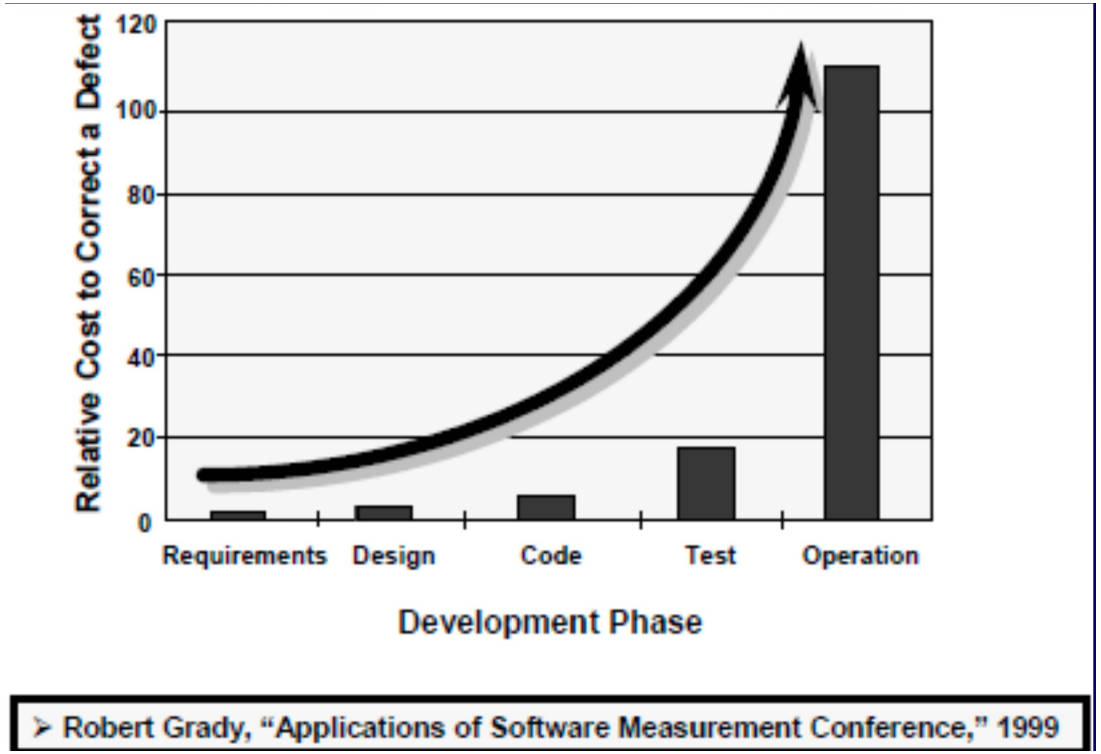
만약 어느 한 단계를 대충한다면?!

혹은 뭔가를 다시 고쳐야한다면?!

Source: Construction Building Worker - Free photo on Pixabay - Pixabay

소프트웨어도 마찬가지입니다.

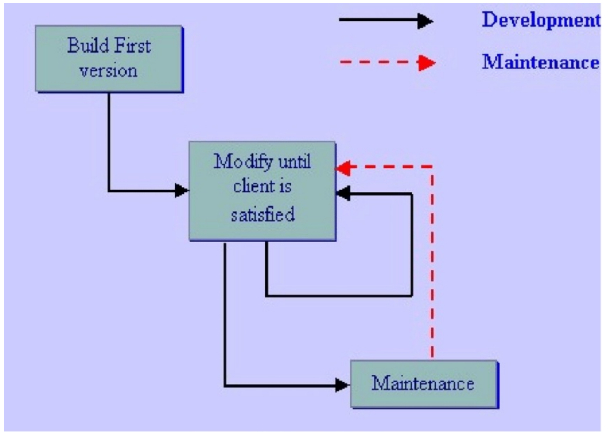
소프트웨어 출시 후 발견된 요구사항 오류는 그 수정 비용이 매우 높지만, 초기에 발견된다면 그 비용이 매우 낮아집니다.



➢ Robert Grady, "Applications of Software Measurement Conference," 1999

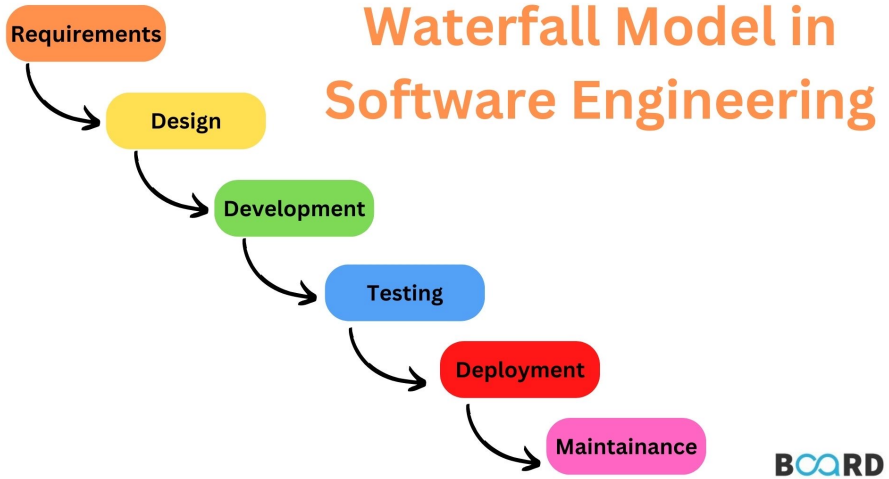
그래서 " 개발 방법론"이 있습니다.

소프트웨어 개발 과정은 일련의 틀(Framework)에 맞춰 진행됩니다.



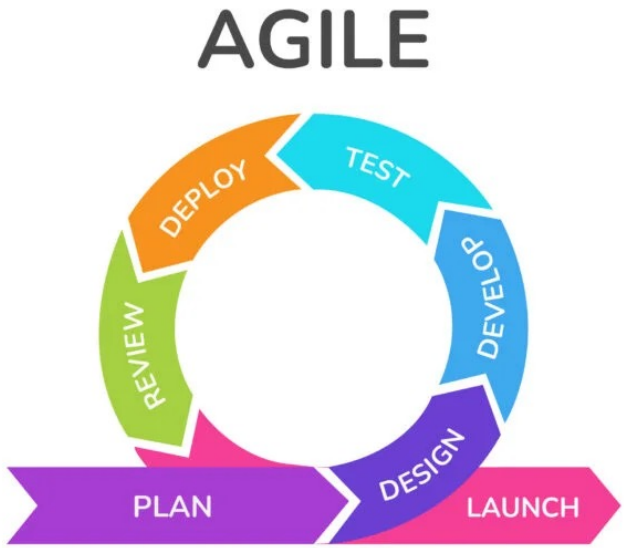
Build & Fix Model

Source: The build-and-fix model | Download Scientific Diagram



Waterfall Model

Source: Waterfall Model in Software Engineering | Board Infinity



Iteration Model

Source: The Agile Development Process for Mobile Apps | Krasamo

여기서의 공통점은?

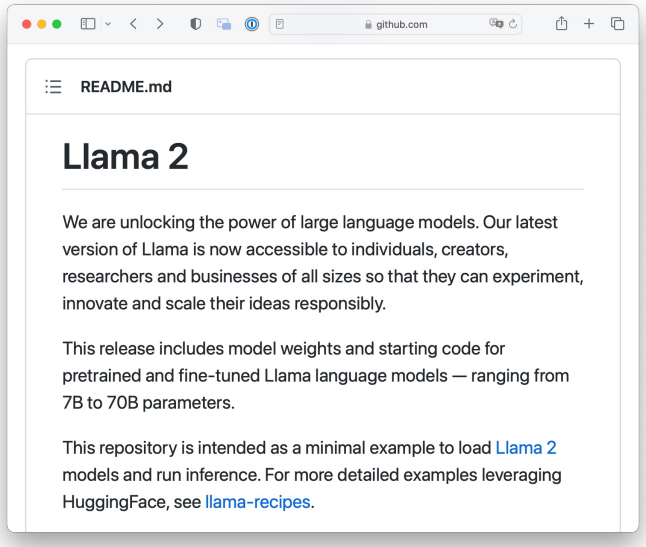
요구사항!

주의) 기술적 세부사항들도 필요하지만, 비기술적인 언어로 소프트웨어가 어떤 모습이 되어야 하는지 기술하는 것이 필요합니다.

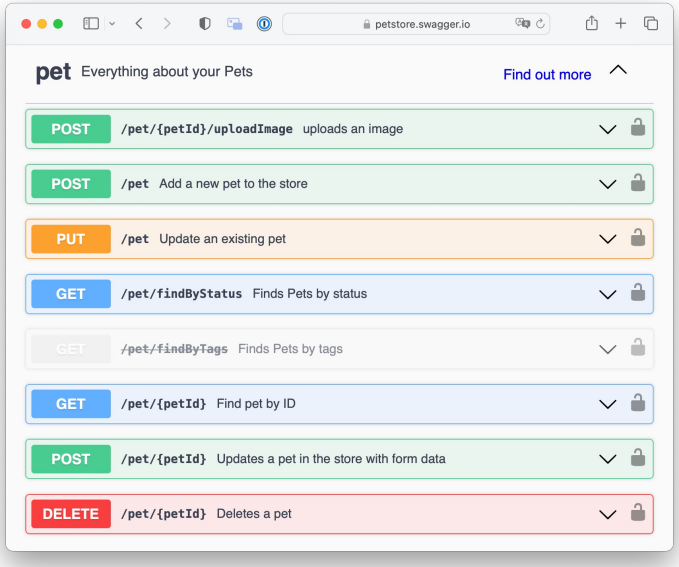
그리고 이 요구사항들을 정리하는 문서를 만드는게 필요합니다.

개발 중일 때

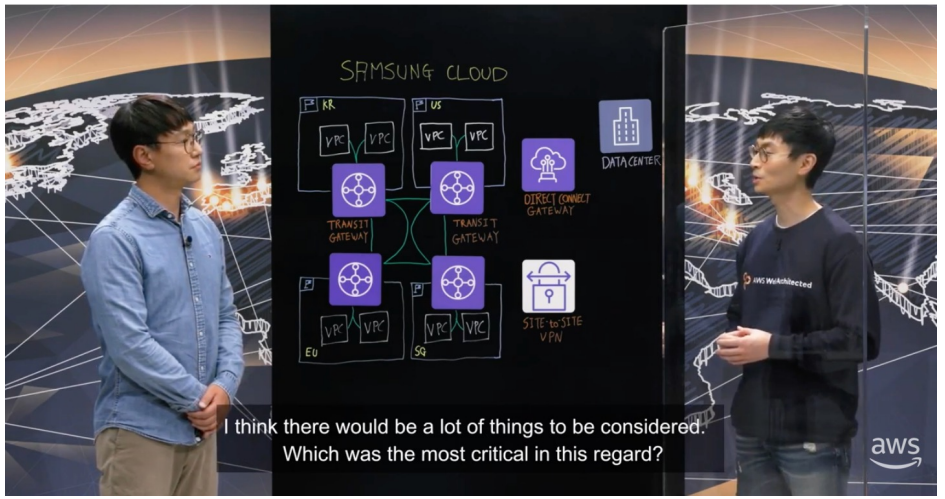
소프트웨어의 구현 현황을 알 수 있는 방법이 필요합니다.



Source: GitHub - facebookresearch/llama: Inference code for LLaMA models



Source: Swagger UI



Source: Samsung Cloud: Global Hybrid Network Optimization Across 5 AWS Regions Using AWS Transit Gateway - YouTube

정리

소프트웨어 개발은 건축 현장과 유사하게 정해진 체계를 따라 진행됩니다.

- 이 때 프로그램의 요구사항과 같은 다양한 사항들이 결정되고
- 개발 과정의 현황을 파악할 수 있게 하고
- 추후 그 결과물들을 설명할 수 있게 하는 문서(또는 그에 준하는 흔적)가 필요합니다.

쉬는 시간

11시에 다시 시작합니다.

요구사항 명세서 (SRS)

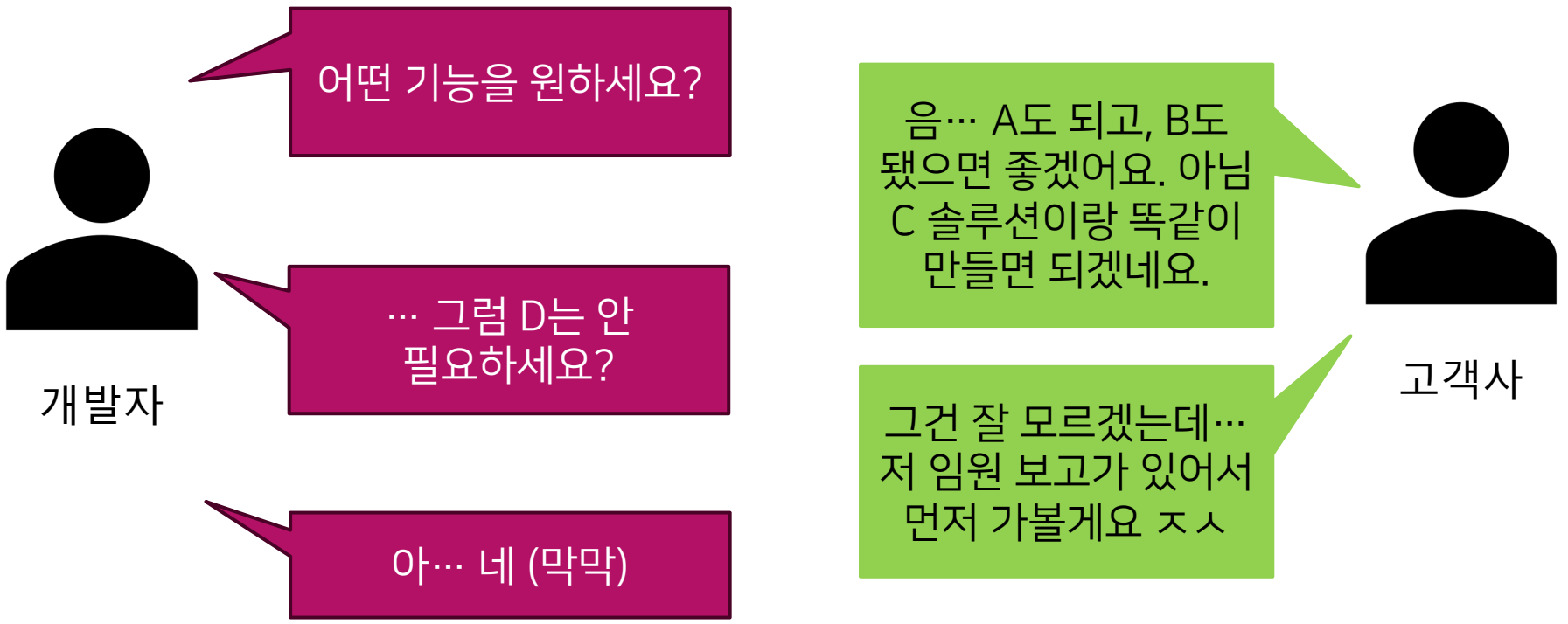
Source: "IEEE Recommended Practice for Software Requirements Specifications," in IEEE Std 830-1998 , vol., no., pp.1-40, 20 Oct. 1998, doi: 10.1109/IEEESTD.1998.88286.

들어가기에 앞서

요구사항 명세서(SRS)에 대해 소개하는 것은 여러분이 이것을 머릿속에 꼭 넣어둬야 한다는 것은 아닙니다!

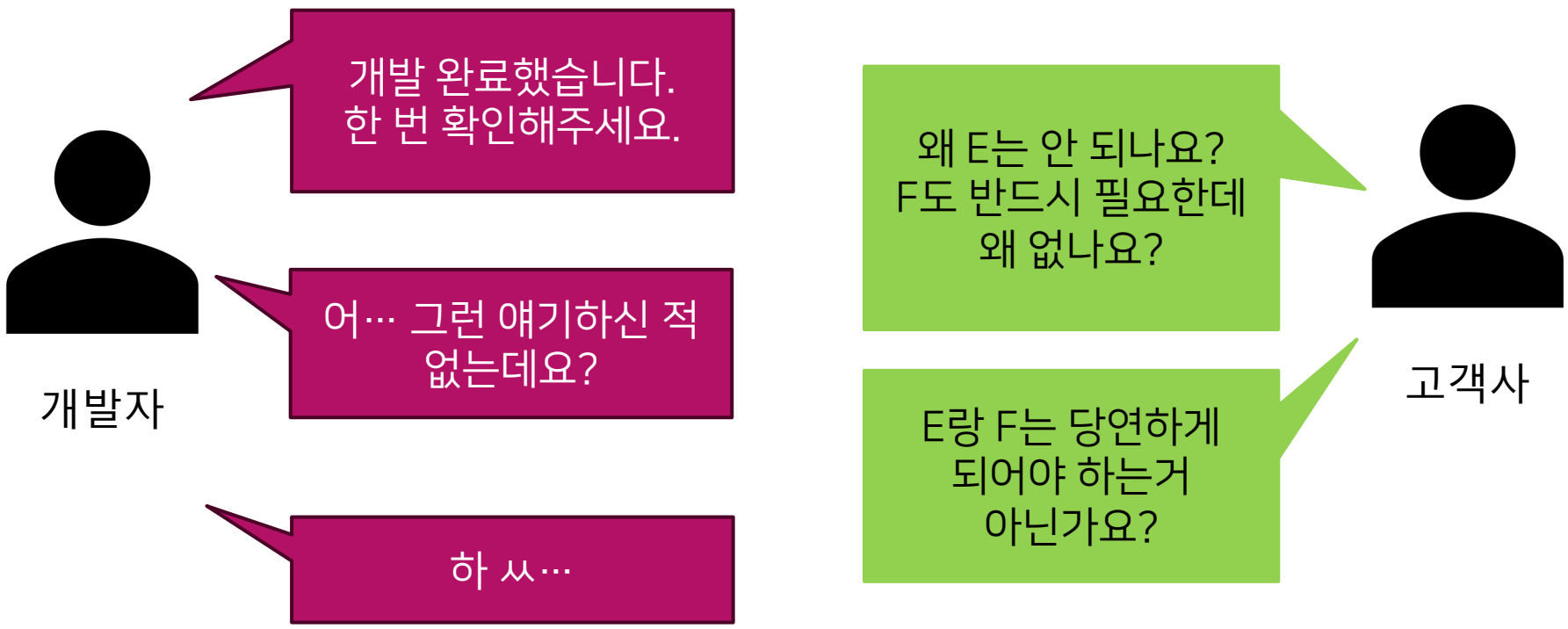
- 소프트웨어 개발을 할 때 생각해야할 항목들을 대략적으로 알고
- 이들을 어떤 식으로 작성해야 하는지 감을 잡는 정도로 알아두시면 좋겠습니다.

근본이 없던(?) 시절 – 개발 초기



유의사항: 극단적인 예시입니다.

근본이 없던(?) 시절 – 개발 후



유의사항: 극단적인 예시입니다.

Software Requirement Specification

소프트웨어의 요구사항을 정리할 때 필요한 항목들과 권장 작성 방식을 기술한 표준 문서

- 소프트웨어 제공자와 사용자가 소프트웨어에 대한 기대치를 서로 맞출 수 있게 하고 (Aggreement)
- 개발 공수(Development Effort)를 줄여줄 수 있으며
- 비용(Cost)과 일정(Schedules)을 산정하는 기초 자료를 제공하고
- 검증 (Validation & Verification)과 개선(Enhancement) 기반을 마련해줍니다.

필요 항목

SRS는 크게 5가지 항목에 대한 설명들이 담기는 것을 권장하고 있습니다.

- **기능** 소프트웨어가 무슨 일을 해야 하나?
- **외부 인터페이스** 소프트웨어는 사용자와 어떻게 상호작용 해야 하나? (혹은 유관 시스템 등)
- **성능** 소프트웨어가 얼마나 잘 작동해야 하나? (응답 시간, 고가용성, 복구 전략 등)
- **비기능적 속성** 그 외에 소프트웨어가 갖춰야할 요건들이 있을까? (보안, 유지보수성 등)
- **제약사항** 개발할 때 꼭 고려해야할 걸림돌은 없을까?

작성 원칙

SRS는

- Correct - 모두가 동의해야 하고
- Unambiguous - (애매모호하지 않게) 명료해야 하며
- Complete – 모든 요구사항들이 담겨야 하고
- Consistent – 그 내용들은 일관성이 있어야 하며
- Ranked – 중요도나 안정성에 따라 순위가 매겨져 있어야 하고
- Verifiable – 그 결과를 검증할 수 있어야 하며
- Modifiable – 수정할 수 있고
- Traceable – 추적 가능해야 합니다.

이 원칙은 SRS에 국한하지 않고
대부분의 기술 문서를 작성할 때 적용할
수 있습니다.

SRS 예시

- 원문 – 링크
- 번역본 (DeepL 자동 번역) - 링크

정리

- 소프트웨어 개발에 앞서 이해관계자들이 협의하여 정의한 소프트웨어의 각종 특성을 기록하는 것이 필요합니다.
- 이 때 소프트웨어 명세서(SRS)를 작성하여 활용할 수 있으며,
- SRS의 내용은 합의된 모든 사항들을 포함하여 명확한 표현으로 작성되어, 추후 검증 가능한 형태로 작성이 되어야 합니다.

쉬는 시간

1시에 다시 시작합니다. 점심 맛있게 드세요.

다이어그램

다이어그램?

다이어그램은 소프트웨어의 계획이나 상황을 시각적으로 보여주는 수단입니다.

다이어그램은 어떻게 구성되어야 할까?

- 서로 다양한 이해당사자들과 (명확하게) 소통할 수 있는 수단이 되어야 하고
- 특정 기술에 종속되지 않아야 하며
- 문제를 해결하기 위한 방법을 제시할 수 있어야 합니다.

다이어그램?

예) UML 다이어그램

기본적 구성 요소

- 어떤 것들이 있고(Thing)
- 어떤 관계를 맺고 있으며(Relationship)
- 그것들을 시각적으로 표현합니다.(Diagram)

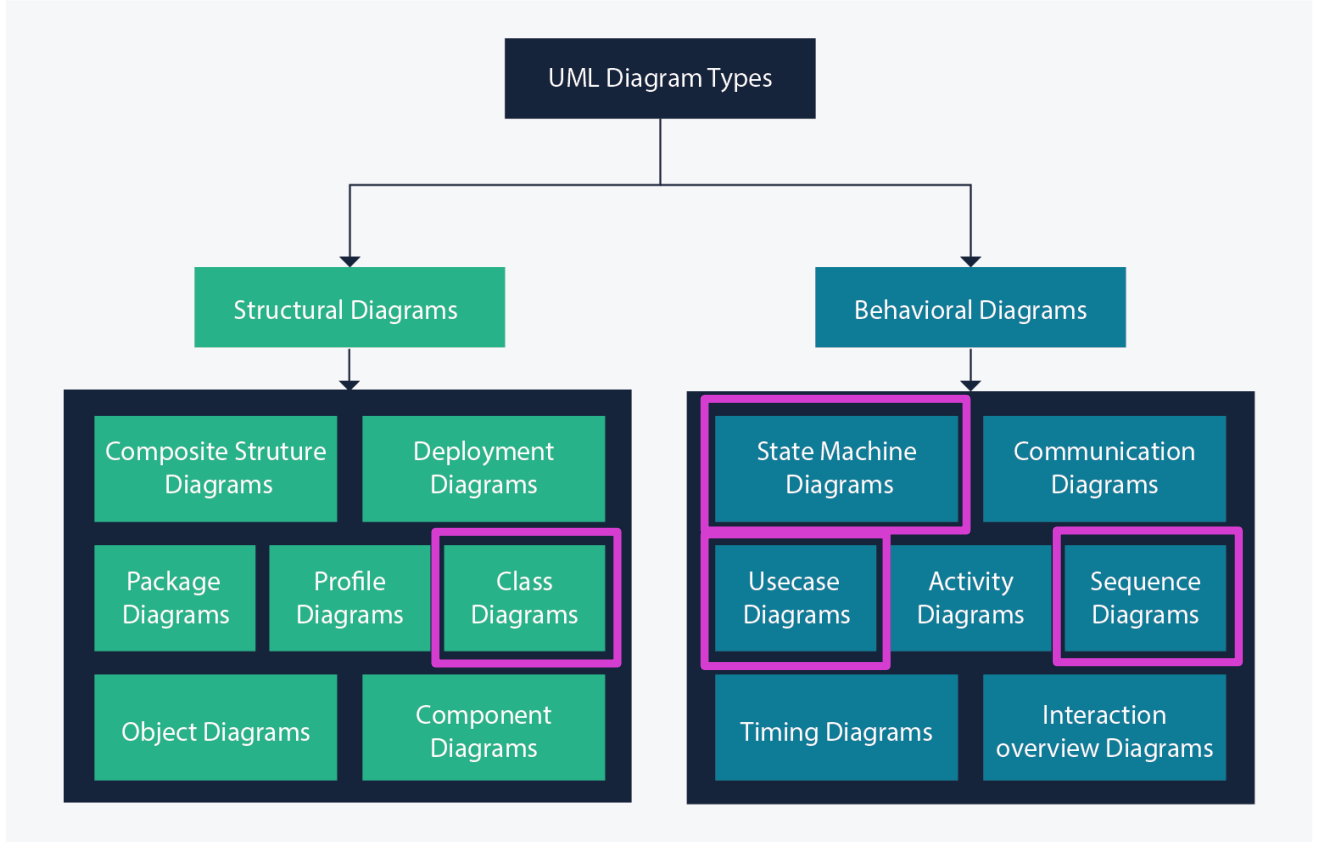
UML 다이어그램의 (구체적) 목표

- 구조를 도식화하고(Structual)
- 내부에서 일어나는 행동들을 도출하며(Behavioral)
- 관련있는 것들을 묶고(Grouping)
- 필요한 내용들을 적습니다.(Annotate)

UML 다이어그램의 종류

Structural Diagram "구조"를 표현하는데 활용

Behavioral Diagram "동작"을 표현하는데 활용



Source: UML Diagram Types | Learn About All 14 Types of UML Diagrams

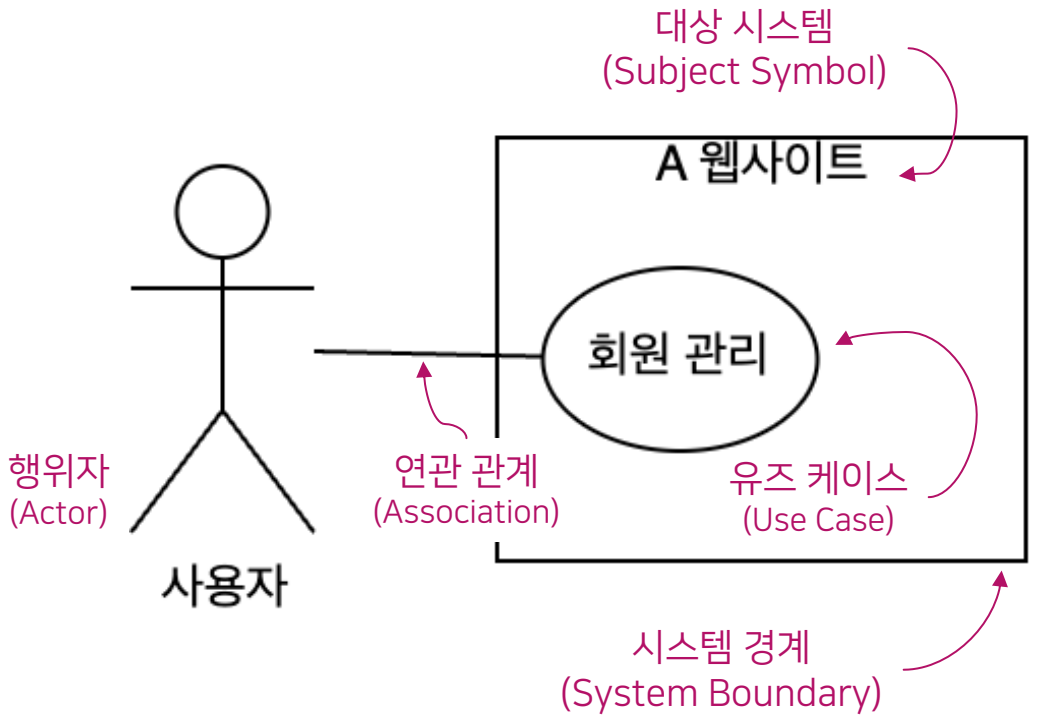
Use Case Diagram

Use Case?

- 행위자가 요구하여 시스템이 수행하는 일의 목표

Use Case Diagram 구성 요소

- 이 외에도 유즈 케이스 간의 포함 관계를 나타내는 종속 관계(Dependency)도 존재합니다.



실습) Use Case Diagram

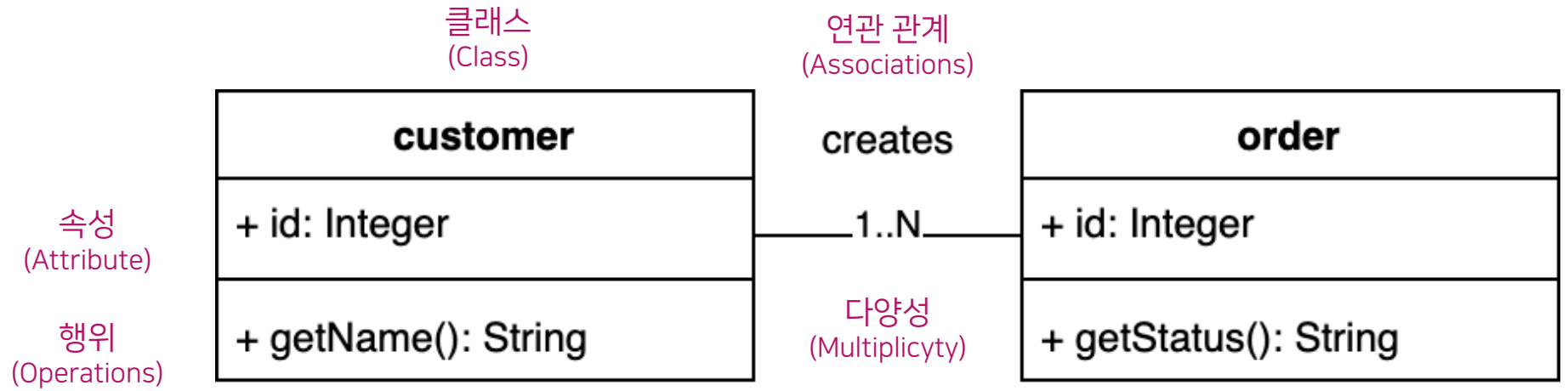
반려동물 가게(a.k.a. Petstore)의 웹 사이트에 대한 Use Case Diagram을 같이 그려봅시다.

- 고객은 우리 웹사이트를 통해 분양받을 수 있는 반려동물들의 목록을 확인하고, 원하는 반려동물 분양을 신청할 수 있습니다. 또한, 신청한 분양 건의 상태를 확인할 수 있으며 자신의 회원 정보도 수정할 수 있습니다.
- 가게 주인은 웹사이트를 통해 들어온 분양 요청을 확인해 수용 가능한 분양 건은 승인할 수 있으며, 가게 운영에 대한 전반적인 재무 상황을 확인할 수 있습니다.

Class Diagram

관련있는 정보들과 행위들을 한데 묶어 구조화한 결과를 표현하는 다이어그램

- 참고) Entity Diagram



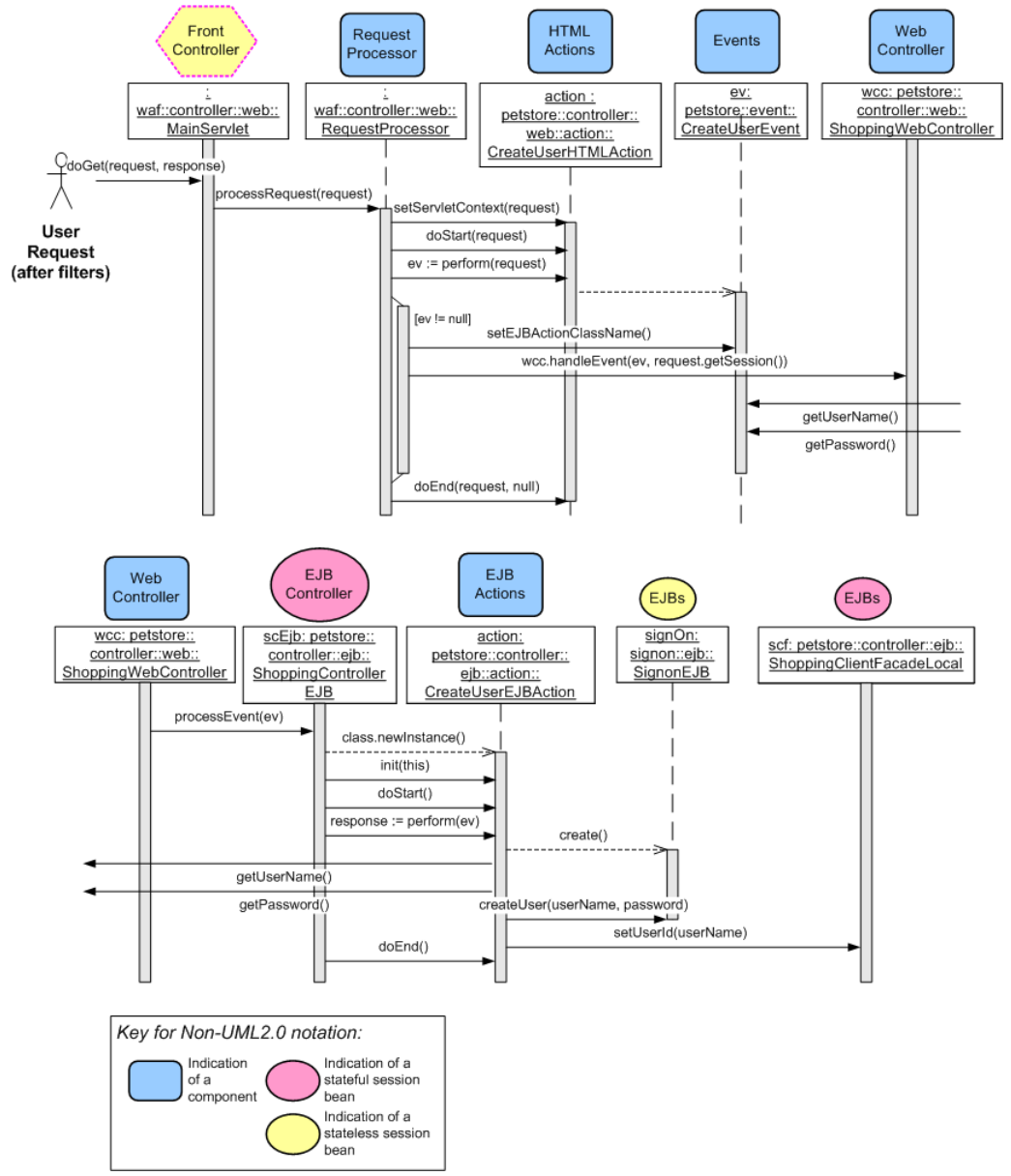
실습) Class Diagram

반려동물 가게(a.k.a. Petstore)의 웹 사이트에 대한 Use Case Diagram을 같이 그려봅시다.

- 웹 사이트의 동작을 위해 일련의 정보가 필요합니다.
  - 고객의 ID, 이름, 이메일 주소, 비밀번호를 필요로 합니다.
  - 반려동물은 반려동물의 종류, 사진(URL), 현 분양 상태, 그리고 기타 특성들을 알 수 있는 태그가 필요합니다.
  - 분양 신청의 경우 분양될 반려 동물이 어떤 것인지, 언제 분양이 완료될지, 그리고 분양 상태가 필요합니다.
- 웹 사이트의 클래스들은 유즈 케이스로 명시된 행위들을 수행할 수 있어야 합니다.

참고) Sequence Diagram

소프트웨어의 서로 다른 (내부) 구성 요소 간의 상호작용 순서를 나타내는 다이어그램



Source: Petstore Application - Software Architecture Documentation (SAD) - Confluence

쉬는 시간

2시에 다시 시작합니다.

준비사항) 여러분의 Django 프로젝트 환경을 준비해주시고, Spinx를 설치해주세요.

```
pip install sphinx sphinx_rtd_theme
```

도구를 사용한 자동화

문서 작성이 필요하긴 한데…

앞서 다양한 문서(혹은 다이어그램)들을 접하긴 했지만, 작성해야 할 문서들은 더 있습니다.

- 웹 서비스라면 API Specification
- 데이터베이스를 사용한다면 ERD(Entity Relationship Diagram)
- SDK를 제공해야 한다면 그에 대한 문서까지…

이걸 언제 다 만들고 있지?!

자동화를 합시다!

문서와 다이어그램들을 작성하는 일부 과정은 어느 정도 정형화가 되어 있어서 기계의 힘을 빌릴 수 있습니다.

- 자동화를 하는 도구는 다양하고, 이를 활용하는 방법은 더 다양합니다.
- 따라서 본 시간에는 자동화할 수 있는 문서화 과정 중 대표적인 것들을
  - ERD Diagram (DBeaver)
  - API Specification (Swagger)
  - Code Documentation (Sphinx)

실습) ERD Diagram

Database 관리 도구를 활용한 ERD Diagrram 생성

- feat. SQL Generation

실습) Code Documentation

Sphinx를 활용해 여러분이 작성한 Python 코드의 문서를 생성합니다. (feat. docstring)

실습) API Documentation

Swagger를 통해 여러분의 Django 프로젝트의 API들을 OpenAPI Specification 형태로 남겨봅시다.

쉬는 시간

3시에 다시 시작합니다. 점심 맛있게 드세요.

문서화 실습

문서화 실습

오늘 배운 내용을 종합적으로 정리하기 위해 여러분이 진행한 프로젝트 중 하나를 선정해 이에 대한 기술 문서들을 만들어봅니다. **(팀별로 진행합니다.)**

- SRS는 필수로 작성하고, 이외의 문서 및 다이어그램들은 시간 여력에 따라 선택적으로 진행합니다.
- 얼마나 많이 만드는지, 그리고 그 퀄리티는 너무 신경쓰지 마세요.
- 실습은 15:00 ~ 16:50까지 진행하고, 마지막 시간은 각자 발표를 진행할 예정입니다.

문서화 실습

많이 막막할 수 있습니다. (예. 뭔가 뻥 뚫려있어요! 어떻게 적어야할지 모르겠어요! 등)

- 막막할 때는 저와 함께 (나름의) 답을 함께 찾아가보면 되니 저를 불러주세요.
  - #라운지-강의중 텍스트 채널에 질문을 올려주시거나 각 팀 채널에서 멘션 걸어주세요.

참고) SRS는 2교시에 보여드린 예시의 항목들을 위주로 작성하시면 됩니다.

실습시간

17시에 다시 모입니다.

결과물 발표

끝내기 전에...

" 글쓰기는 여러분의 생각을 정리하는데 도움이 됩니다."

어떤 이유에서인지 우리 인간은 지식이 곧 독서와 같다고 생각합니다. Rust 시작 가이드를 읽으면 Rust에 대해 안다고 봅니다. TCP/IP가 어떻게 작동하는지에 대한 책을 읽었으니 이제 TCP/IP에 대해 안다고 봅니다. 하지만 사실이 아닙니다. 사실이라면 우리 모두 천재가 되었을 것입니다.

글쓰기는 각자의 지식을 확고히 해줍니다. 그래서 저는 코드 조각을 복사하는 대신 코드를 작성하는 것을 더 선호합니다. 코드를 작성하면 그 내용이 머리에 각인되기 때문입니다.

Source: Why engineers should focus on writing - Dmitry Kudryavtsevl

“글쓰기는 자신의 실수를 찾아내는데 도움이 됩니다.”

디자인 문서를 준비하라는 요청을 받았을 때 디자인이 너무 단순해서 머릿속에 쉽게 들어오는데 왜 작성해야하는지 의문을 가진 적이 있나요? 사실 이는 큰 오해입니다. 실제로 디자인 문서를 작성할 때 ‘단순한' 디자인에서 많은 문제점을 발견했을 것입니다. 내용의 불일치, 누락된 세부 사항, 또는 대충 생각해서 말이 안 되는 것들을 찾곤 합니다.

Source: Why engineers should focus on writing - Dmitry Kudryavtsevl

“(회사 밖에서) 글쓰기 능력을 기르고 싶다면 블로그를 시작해보세요”

블로그 글을 작성하는 것이 부담스럽다면 Stack Overflow의 질문에 답변하는 것을 고려해보세요. 대신 코드 조각을 복사하여 붙여넣기보다는 해답의 맥락을 설명하는 것에 더 신경 써보세요.

마지막으로 한 가지 조언을 드리자면, 무작정 복사/붙여넣기를 하지 마세요. 제가 멘토링한 많은 개발자가 코드 조각, 함수 선언 등을 포함한 모든 것을 복사/붙여넣기 합니다. 저는 Git 저장소를 초기화하는 방법을 매번 직접 하기 때문에 잘 알고 있습니다만 대부분의 사람들은 GitHub나 Google 검색 결과를 가져다가 사용하기만 합니다. 만약 직접 치는 것이 비생산적일까 봐 걱정된다면, 본인이 코드를 얼마나 많이 작성했는지, 얼마나 빨리 과제를 완료했는지에 따라 평가받는 것이 아니라는 점을 기억하세요.

Source: Why engineers should focus on writing - Dmitry Kudryavtsevl